EXHIBIT 5

REDACTED

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF VIRGINIA
ALEXANDRIA DIVISION

UNITED STATES, et al.,	§	
	§	
Plaintiffs,	§	
v.	§	No. 1:23-cv-00108-LMB-JFA
	§	
GOOGLE LLC,	§	
	§	
Defendant.	§	

EXPERT REPORT OF JASON NIEH, PHD

JULY 7, 2025

HIGHLY CONFIDENTIAL – SUBJECT TO PROTECTIVE ORDER

infrastructure to accommodate higher computing and data requirements to support more users and more transactions.[28]

51. One can think of scaling as the ability for a business to increase (and decrease) its capacity for serving customers as the numbers of those customers increase (and decrease). For example, a grocery store may have 10 check-out counters, but only 2 employees scanning customers' groceries when foot traffic at the store is low. But during peak hours when there are many more customers than usual, the grocery store's managers may scale up grocery scanning throughput by assigning 10 employees to all 10 check-out counters to ensure that all customers are satisfactorily serviced in a timely manner. Once the number of customers decreases, management may scale back the number of cashiers so that those freed-up employees can return to other work at the store.

52. Scaling a software system and the infrastructure on which it operates in an optimized way is not a trivial task, yet it is crucial to meet the needs of a growing number of users or traffic.[29] To achieve the necessary scale to support large amounts of traffic and computational processing efficiently and cost effectively, software and hardware rely on a technique known as *horizontal scaling*. Horizontal scaling is rooted in the idea that if SWEs can design a software system that divides its work into separate smaller tasks that can be run on multiple computers at the same time (i.e., in parallel), then the amount of work the software system can do can be scaled by dividing and distributing the work to more and more computers (a "distributed" computer system). Dividing a software system's tasks into smaller pieces means that SWEs can employ many computers working together, to overcome the technical limitations on how fast a single computer can be.

53. Horizontal scaling requires scaling computational processing. For example, a database system can scale to handle more queries for data by architecting the database program so that it can run on many computers and distributing the queries across those computers so they can be processed in parallel.[30] Horizontal scaling typically relies on distributing work

[28] Arnaud Creput Deposition (Equativ), 6/11/25, at 71:13-19. ("[redacted]").

[29] Weinstock, Charles B., and John B. Goodenough, *On System Scalability*, Carnegie Mellon University (March 2006), at v, https://www.researchgate.net/profile/John-Goodenough/publication/238687854_On_System_Scalability/links/545794060cf2cf51648216a1/On-System-Scalability ("A significant number of systems fail in initial use, or even during integration, because factors that have a negligible effect when systems are lightly used have a harmful effect as the level of use increases. This scalability problem (i.e., the inability of a system to accommodate an increased workload) is not new. However, the increasing size (more lines of code, greater number of users, widened scope of demands, and the like) of U.S. Department of Defense systems makes the problem more critical today than in the past.").

[30] Anderson, Erik and Rohit Bansal. "Scale Amazon Redshift to meet high throughput query requirements," Amazon Web Services (Apr. 20, 2022), at 1, https://aws.amazon.com/blogs/big-data/scale-amazon-redshift-

across many (even millions of) inexpensive, standardized, and easily replaceable "commodity" computers.[31] However, this can be challenging to accomplish and require substantial modifications to a software system to enable it to run concurrently on many machines.

54. Horizontal scaling also requires scaling access to data needed to perform the computations. This includes making the data accessible, and accessible quickly, on multiple computers. Two common techniques for scaling access to data are *replication* and *sharding*.

55. Replication involves storing copies, or replicas, of the data on multiple computers (often in different physical locations). For example, a database system that can run on multiple computers could have a copy of all the data on each computer so that it can be accessed "locally." Queries handled by that computer can then be processed locally without needing to retrieve data from another computer, as the latter often takes longer. Replication can also improve system reliability by providing redundant copies of data in case one is corrupted due to hardware failures.[32]

56. While replication is an important technique for scaling, a difficulty with replication is ensuring that the replicas of the data remain *consistent* with one another. Otherwise, running the same task on two different computers may produce different results because they use different versions of the data.[33] This requires an effective mechanism to allow any changes made to a replica on one computer to be propagated to replicas on other computers. For example, if a withdrawal request reduces the user's account balance and is processed on one computer, it is essential that this update to the user's account be propagated to any

---

to-meet-high-throughput-query-requirements ("As adoption increases and more users need access to the data, you can add nodes of the same node type to your cluster to increase the amount of compute power available to handle those queries. An elastic resize is the fastest way to horizontally scale your cluster to add nodes as a consistent load increases.").

[31] Using a large number of commodity computers (which are inexpensive and easily replaceable) is an alternative to using a smaller number of high performance computers that are expensive and sometimes difficult to maintain. Not only is the latter approach more expensive, but in many cases it is faster to perform many more parallel, distributed computations on the commodity computers compared to using fewer expensive, high-performance computers with less parallelism. Parallelism in computer science and software engineering refers to the extent that a software is designed and written to strategically divide work into smaller parts that can be computed simultaneously, i.e. "in parallel." Software with a high degree of parallelism is naturally highly suited to horizontally scaling infrastructure as each small computational piece of a larger task can be delegated to a commodity computer.

[32] Barroso, Luiz André, et al., *The Datacenter as a Computer: Designing Warehouse-Scale Machines*, Springer International Publishing, (3d ed. 2022), at 68 ("Cross-machine replication allows the system to tolerate machine and network failures and enables fast recovery, since replicas for a given disk or machine can be spread across thousands of other machines.").

[33] Barroso, Luiz André, et al., *The Datacenter as a Computer: Designing Warehouse-Scale Machines*, Springer International Publishing, (3d ed. 2022), at 34 & Table 2.2 ("Data replication can improve both throughput and availability. It is particularly powerful when the replicated data is not often modified, since replication makes updates more complex.").

replicated record of the user's account on other computers before responding to any requests for that balance on other computers. A system is said to be *strongly consistent* if it returns the same data in response to a request regardless of which computer is used to process the request.[34]

57. Sharding involves partitioning the data across multiple computers (often in different physical locations) based on the tasks the computers perform. In other words, each computer is only given the portion of the data it needs to perform the operations with which it is tasked. Sharding may be necessary because there is too much data to be stored on any one computer, so it must be divided among multiple computers. A difficulty with sharding is deciding how to divide tasks and data among computers so that each task mostly needs only the shards of data stored locally. If this is not done properly and a computer has to frequently access shards of data stored on other computers, then the system can become inefficient in accessing data, limiting scalability and degrading performance and the quality of the software's services for its users.

58. Enabling scalability is particularly challenging because SWEs frequently need to scale software applications while maintaining minimum acceptable levels of performance and preventing slowdowns or failures. In some cases, there may be explicit Service Level Agreements ("SLAs") and Service Level Objectives ("SLOs") that must be met that specify the performance and service level of the software application (and the metrics by which they are measured) as agreed between an application provider and the user (or customer).[35] Failure by the provider to maintain the agreed SLA minimum performance or quality of service levels typically results in financial penalties for the provider. For example, an SLA may specify the maximum response time a software system must provide to its users or the maximum time for which a system can be unavailable due to an outage, thresholds which if exceeded would require the provider to pay the user some form of compensation.[36] To meet

[34] Tanenbaum, Andrew and Maarten van Steen, *Distributed Systems: Principles and Paradigms*, Prentice Hall (2d ed. 2007), at 273-274. ("... In most cases, applications require a strong form of consistency. Informally, this means that updates are to be propagated more or less immediately between replicas. There are various alter/natives for implementing strong consistency, which are discussed in a separate section.")

[35] One challenge to scalability is the need to balance specific performance and service level metrics that are often at odds with each other. For example, increasing queries per second (a measure of throughput; higher is better) can also cause an increase in latency (lower is better). *See* Beyer, B., et al., eds., "Service Level Objectives," *Site Reliability Engineering: How Google Runs Production Systems*, (O'Reilly 2016), https://sre.google/sre-book/service-level-objectives/ (accessed June 26, 2024) ("Again, this is more subtle than it might at first appear, in that those two SLIs [Service Level Indicators]—QPS and latency—might be connected behind the scenes: higher QPS often leads to larger latencies, and it's common for services to have a performance cliff beyond some load threshold.").

[36] *See, e.g.*, Google, Inc., "Cloud Observability (Monitoring, Logging) Service Level Agreement (SLA)," https://cloud.google.com/operations/sla (accessed June 25, 2024) ("During the Term of the agreement under which Google has agreed to provide Google Cloud Platform to Customer (as applicable, the "Agreement"), the Covered Service will provide a Monthly Uptime Percentage to Customer of at least 99.95% (the "Service Level Objective" or "SLO."). If Google does not meet the SLO, and if Customer meets its obligations under

SLAs, the ability to manage computer hardware failures is particularly crucial as they are a daily common occurrence for large-scale distributed computing systems (*see* below for a discussion of how complex computing systems account for failures). In other words, SWEs must scale systems while maintaining minimum levels of performance and preventing slowdowns or failures.

59. Scaling to support large numbers of users and transactions around the world can require not just a distributed computer infrastructure operating in a single data center, but a computing infrastructure that is distributed worldwide. This is for three main reasons: latency, reliability, and legal compliance.

60. Latency is the time it takes for a system to respond to a request made of the system.[37] There are many types of latency. For example, the time it takes for a server hosting a web page to respond to a request to display that web page is considered to be network latency (a delay caused by characteristics of the network). If the server hosting the web page is a long distance physically from the user, that leads to increased latency,[38] and potentially a negative

this SLA, Customer will be eligible to receive the Financial Credits described below.").

See also Google, Inc., "Google Marketing Platform - GA 360 Suite Service Level Agreements" (Dec. 15, 2022),, https://marketingplatform.google.com/about/analytics_products/sla.

See also GOOG-AT-MDL-B-009834156, at -193 (Google Ad Manager Service Order Form which states: "Solely with respect to Traditional Ad Serving, Google will use commercially reasonable efforts to ensure that Traditional Ad Serving processes Ad requests at least 99.5% of the time, calculated on a calendar monthly basis as measured by Google from the data center used by Google to serve Ads on Company's behalf, it being understood that "down" time (calculated as the difference between 100% of time in a calendar month and the actual percentage of time during that month that Ad requests are processed) will exclude time resulting from technical malfunctions in the Target Properties' systems, or any other circumstances beyond Google's reasonable control (including, without limitation, Internet delays, network congestion and ISP malfunctions).").

[37] Beyer, B., et al., eds., "Service Level Objectives," *Site Reliability Engineering: How Google Runs Production Systems*, (O'Reilly 2016), https://sre.google/sre-book/service-level-objectives/ (accessed June 26, 2024) ("Most services consider request latency—how long it takes to return a response to a request—as a key SLI.").

[38] The absolute upper-bound for the speed of data transmission is the speed of light, and even light takes about 133 milliseconds to circle around the globe once. *See* NASA Glenn Learning Technologies Project, "How 'Fast' is the Speed of Light?," https://www.grc.nasa.gov/www/k-12/Numbers/Math/Mathematical_Thinking/how_fast_is_the_speed.htm (accessed June 12th, 2025) ("Light travels at a constant, finite speed of 186,000 mi/sec. A traveler, moving at the speed of light, would circum-navigate the equator approximately 7.5 times in one second." In other words, light would take approximately 133.33 milliseconds to circum-navigate the equator once."). In practice, data transmission is much slower than the speed of light, and thus data can take hundreds of milliseconds (or even longer) to traverse across continents.

See also Cloudflare, "What is latency?," https://www.cloudflare.com/learning/performance/glossary/what-is-latency/ (accessed June 12th, 2025) ("One of the principal causes of network latency is distance, specifically the distance between client devices making requests and the servers responding to those requests. If a website is hosted in a data center in Columbus, Ohio, it will receive requests fairly quickly from users in Cincinnati (about 100 miles away), likely within 5-10 milliseconds. On the other hand, requests from users in Los Angeles (about 2,200 miles away) will take longer to arrive, closer to 40-50 milliseconds.").

financial impact on any implicated business transactions.[39,40] To reduce latency and improve user experience, it is desirable to direct user requests to servers that are physically closer to the users. As a result, a large-scale software system that needs to support users all over the world will have servers running the software in diverse geographic regions and direct user requests in a geographic region to servers in the same or nearby geographic region to reduce latency.[41] In the context of globally-distributed, large-scale software systems, latency is often measured in milliseconds.[42]

61. Reliability can be thought of as how often the system responds to requests or fails to do so. Distributed computing systems build redundancy into the system, i.e. if one part of the infrastructure fails, other parts of the infrastructure can take over.[43] For example, a geographically distributed infrastructure provides protection against a single regional event (e.g., a weather event) causing the entire system to fail: if the infrastructure in one location (e.g., a data center) fails, infrastructure in other locations that have replicas of the required data can take over. Thus, a geographically distributed infrastructure can increase the reliability of the system.[44] Reliability is also important not just for catastrophic failures such

[39] *See generally* Konstantin Mirin, "Header Bidding: Definition, Pros & Cons," Post Industria (Jul. 8, 2020), https://postindustria.com/header-bidding-definition-pros-cons-adtech (discussing header bidding causing latency and lost profits.).

[40] Glenn Berntson Deposition, 6/24/25, at 16:9-22:12 ("When you introduce latency into the system, the ads don't show up right away, and that delay can have a huge impact on whether or not a publisher is earning money by the attempt to deliver ads to their site.").

[41] Cloudflare, "What is latency?," https://www.cloudflare.com/learning/performance/glossary/what-is-latency/ (accessed June 12th, 2025) ("Use of a CDN (content delivery network) is a major step towards reducing latency. A CDN caches static content and serves it to users … CDN servers are distributed in multiple locations so that content is stored closer to end users and does not need to travel as far to reach them.").

[42] For example, Google's AdSpam Real Time Annotation Service ("RTAS") budgets response times that are generally within 100 milliseconds ("ms") for the many Google Ad Tech systems that depend on it. *See* GOOG-AT-MDL-B-009826812, at -821 (2023 Google presentation on RTAS identifying response time budgets of 20ms for CAT2, 100ms for Click Logger, 10ms for Xbid, and 20ms for DFP).

[43] Nitish Korula Deposition, 6/9/25, at 122:14-124:19 ("For a specific hardware upgrade, essentially what you would do in the data center and infrastructure teams with Google is require the system to be built with spare capacity and monitoring to ensure that even if any data center were to be taken offline for any reason at any time, that the service would continue to run. So when building the service, you have to build it in same exact what. And so then, when the data center operations team would sort of like go in and take a data center offline because they needed to make changes or upgrades or anything, we would automatically, you know, serve ad requests from a different data center, because we had designed it in this [] sort of way.").

[44] Google Cloud Architecture Center, "Building blocks of reliability in Google Cloud," https://cloud.google.com/architecture/infra-reliability-guide/building-blocks (accessed June 12th, 2025) ("Google Cloud infrastructure services run in locations around the globe. The locations are divided into failure domains called regions and zones, which are the foundational building blocks for designing reliable infrastructure for your cloud workloads. … Google Cloud infrastructure is designed to tolerate and recover from failures. … Geographically separated regions [] mitigate the effects of natural disasters and region outages on global services. …").

needed to invest time in building new, innovative features for the product as that work was ongoing.[377]

I further note that assuming the availability of up to 300 system-qualified engineers for 5 or more years to work on divestiture may not be realistic under the circumstances because, as a practical matter, even normal attrition—to say nothing of attrition from not working on product innovation, but creating a copy of an existing product[378]—during the lifespan of the divestiture project may reduce the already-limited number of engineers with the necessary technical expertise and domain knowledge of Google's systems.[379,380]

c. The target infrastructure that a buyer would use (e.g., provisioning infrastructure from AWS versus building a data center from scratch) would depend on the buyer's choices,[381,382] and therefore the identification of a buyer is a necessary prerequisite for all the required steps to divest AdX or DFP. Put another way, the

[377] George Levitte Deposition, 6/13/25, at 69:17-70:10 ("[redacted]).

[378] George Levitte Deposition, 6/13/25, at 57:2-58:17 ([redacted]).

[379] GOOG-AT-MDL-004122442 at -448 ("Google has acquired more than 150 companies to date. Below are the attrition rates of our largest acquisitions: …"), -450 ("33% of key employees (n=463) have voluntarily attrited since their acquisition, significantly higher attrition among all other acquired Googlers (27%)."). Note that the 33% and 27% rates of attrition only include cases of voluntary attrition, and do not account for involuntary attrition, which occurred in nearly all acquisitions sampled at page -448. Acquired Project Managers (PMs) have an even higher attrition rate of 41% within 60 months of acquisition. GOOG-AT-MDL-004122442 at -453.

[380] George Levitte, 6/13/25, at 57:2-58:17 [redacted]).

[381] Nitish Korula Deposition, 6/9/25, at 153:5-155:7 ("[T]o answer your question at all … I would do the thing that makes more sense, which is to start an entirely new system that doesn't have any difference to the Google system. Now here, it depends on what is available to me, right. And, therefore, it also depends on who the buyer is … Depending on who the buyer is, sort of their own internal systems are going to be, you know, very different.").

[382] George Levitte, 6/13/25, at 102:22-104:2 ([redacted]").

remaining steps could not commence until a buyer is identified. For purposes of my discussion here, I assume as a starting point that a buyer has already been identified, and thus my opinions regarding the overall timeline for completing the divestiture of AdX or DFP follow any period of time required to identify a buyer.

d. The discussion below generally assumes that the process would be a collaborative effort between Google (which will have the most knowledge about the existing systems and technology) and the divestiture buyer (which will have the most knowledge of its plans for the divested product, the environment in which it will operate, etc.).[383] The details of that collaboration, and the negotiation of the scope of each party's responsibilities, are largely beyond the scope of my report, and I do not address any potential barriers to the sharing of information (including Google's non-ad tech specific trade secrets) that could complicate the migration of DFP or AdX to a new infrastructure.

158. Ultimately, it is extremely difficult to plan and estimate timelines for a project involving this many dependencies and such complex codebases.[384,385] Estimating timelines for software engineering projects requires a holistic approach that accounts for planning, infrastructure provisioning, and time spent coding, testing, and optimizing the system—all critical steps described below. But many of these steps are inherently unpredictable because they are so

[383] Jay Friedman Deposition (Goodway Group), 6/30/25, at 94:20-95:7 ("[redacted]).

[384] Even relatively short one-year projects introduce enough uncertainty that it is typical to multiply an initial estimate by 1.5 to 3 times to account for unexpected outcomes. *See* Nitish Korula Deposition, 6/9/25, at 287:5-288:1 ("Q. But when engineers working in the ads group came up with estimates for projects, would this be a typical format in which the estimates were presented, in other words, number of engineers and how much time? … A. Yes, generally, in terms of number of engineers and time. … Typically, depending on the nature of the project, there might be more uncertainty. . . When you're estimating something that is a year long, it's likely that that's an imprecise estimate, and [redacted]").

[385] George Levitte Deposition, 6/13/25, at 82:1-18 ("[redacted]").

complex.[386] The estimates I present in this section therefore represent at best an optimistic approximation of timing even if everything went exactly according to plan, which is itself an unrealistic assumption given the uncertainties inherent in significant software engineering projects.

1. Creating an Architectural Development Plan

159. The first step for a divestiture buyer would be to plan how to set up the divested versions of AdX or DFP so that they can operate independently of Google in the buyer's existing or to-be-established infrastructure ecosystem. Based on the requirements of the AdX and DFP functionalities (just a subset of which were outlined above) and the complex, sophisticated hardware and software infrastructure on which they operate, I expect producing an architecture development plan would take a considerable amount of time,[387,388] and would likely need to be revisited and revised many times during the divestiture process due to changing conditions and requirements.[389]

[386] Even in the ordinary course of business, Google has observed that migrating software across infrastructures can be challenging and create significant uncertainty. For example, Google acquired Waze, a mapping software company, in 2013. [redacted] GOOG-AT-MDL-B-009835920, at -921 ("[redacted]"). As demonstrated by an internal document, [redacted] GOOG-AT-MDL-B-009835920, at -931 ("[redacted]"); -942 ([redacted]").

[387] George Levitte, 6/13/25, at 82:1-18 ([redacted]").

[388] [redacted] Andrew Casale Deposition (Index Exchange), 7/3/25 at 100:17-101:6 ("[redacted] 102:14-20 ("[redacted]

[389] Nitish Korula Deposition 6/9/25, at 108:10-109:21 ("[T]his is often the case on software systems, right, depending on what you know you're building, the design isn't complete in a separate phase, right. Like, it's a sort of, like, multistage thing, where you design a piece, you deploy it, you design the next piece, you deploy it. You realize something needs to change so you design a modification and deploy that.").

160. Creating the architecture development plan would first entail a review of Google's ad tech systems, and the systems on which they rely (including the shared infrastructure utilized by other Google products), to identify the systems that are necessary to facilitate the ad serving and exchange functionalities.

161. The next step would be to create a work plan for migrating ("porting")[390] over or recreating components of the ad tech stack that facilitate the ad serving or exchange functionalities, as well as a work plan for recreating as needed, outside of Google, the software systems that the ad serving and exchange functionalities rely on and are currently utilized by multiple Google ad tech products. In other words, before the buyer can plan a migration path, it needs to know which pieces of software code it has to migrate and which it has to recreate—which, depending on the buyer, could be nearly everything.[391] As discussed above in Section IV, much of the code that supports the ad serving and exchange capabilities is intertwined, and also used by other Google ad tech products.[392] An architecture development plan would thus require identifying what code is required to run either ad serving or ad exchange functionality on its own, as well as planning for how to modify any relevant code to enable AdX and DFP to run separate from each other.

162. Once SWEs understand which software components would be recreated, they would also need to identify which specific parts of Google's sizable hardware infrastructure GAM relies on would have to be recreated.[393] The challenges of actually setting up such an equivalent hardware infrastructure, once a buyer decides on a strategy, are discussed in more detail below.

163. After a buyer decides on a hardware infrastructure, the buyer must also determine what replacements for the Google proprietary infrastructure software shared across Google products are needed, and define an operating environment for the divested DFP or AdX. This means identifying sufficient replacements (to the extent they exist) for Borg, F1, Spanner, Colossus, logging infrastructure, GFE infrastructure, and the many other custom

[390] Porting is the alteration of software to function in a different environment.

[391] George Levitte, 6/13/25, at 90:17-92:5 ([redacted].").

[392] Glenn Berntson Deposition, 6/24/2025 at 152:1-152:20 ("There are, really, I think, two high-level concepts that are captured by commingling. One is where, when you're actually looking at code, where you've got lines of instructions, that there's some code that exists, say, within one source file that is providing services for, say, AdX functionality or DFP functionality. But in many cases, ultimately, sort of merged so it's the same code that's providing services both to AdX and DFP. Much like we talked about earlier, there's this gray area where GAM represents the fusion of these things that, ten-plus years ago, were distinct products. They are not distinct products anymore.")

[393] *See e.g.*, GOOG-TEX-00054421 at -422 (providing an example of how Google planned the migration of Admeld to Google infrastructure and code) ("Phase 1: Prototyping and Planning: During this phase we will demonstrate that all of the technologies needed to complete a migration are available (or identify and remedy any gaps).").

Google dependencies on which GAM's ad serving and exchange functionalities rely. The process of identifying a sufficient replacement or alternative for Google's custom software is characterized by a high degree of uncertainty because, in general, there are likely no existing technologies with features and capabilities that exactly match with Google's software. Even in the case that an analogous (but not identical) alternative technology exists, work would still need to be done to make the new AdX or DFP's many dependencies compatible with the identified alternative. If no suitable alternative can be identified for a given dependency (including within the divestiture buyer's own proprietary software), that dependency will need to be replicated, and the same process repeated for the dependencies of those dependencies. Such a task would require engineers who are experts in not just the existing Google proprietary software that is to be replaced, but also the possible replacement software operating environments.

164. For example, it is a near certainty that there is no off-the-shelf distributed relational SQL database that supports F1's feature set, performance, and scalability at a global level.[394] Even if the divestiture buyer were another company with hyperscale infrastructure, then it may at best have its own custom, proprietary distributed database that could serve as an alternative to Google's F1. Even so, it is very unlikely that the hyperscaler's F1 analogue would be designed in the exact same way, and significant effort would likely be required to understand the differences and rectify any incompatibilities. For example, a key challenge would be understanding, in great technical detail and not just at a high level, the consistency models supported and their performance implications. The engineering team would then have to determine whether it is better to try to modify the rebuilt AdX or DFP as a whole to use the hyperscaler's alternative to F1, modify the hyperscaler's analog to provide features more similar to F1, or attempt to replicate F1 entirely for use at the hyperscaler. Deciding on strategy to resolve these and similar issues would dominate much of the planning stages before any software engineering work can even begin.

165. At a more general level, the next step for the SWEs planning the divestiture would be to determine how best to migrate AdX or DFP functionalities to a new infrastructure.

166. One approach might be what is referred to as "lift-and-shift," i.e. to attempt to migrate an existing software system to the cloud with as few changes as possible. This approach would entail "lifting" the relevant software code off its existing platform infrastructure and "shifting" it over to the new platform infrastructure to figure out what needs to be done to make the software work on the new platform. As I discuss in the next section, given GAM's substantial dependencies on bespoke Google services, even a "lift-and-shift" would likely involve a combination of three labor-intensive strategies for writing new code: (a) rewriting

[394] Jeff Shute, et al., F1: A Distributed Database that Scales, *Proceedings of the VLDB Endowment*, Volume 6, Issue 11 (August 2013) at 1.

certain software dependencies used by the Google code, so they exist outside of Google's environment and in the buyer's; (b) writing a shim code layer to translate the Google code to work with new dependencies that exist in (or can be created for) the target environment; and/or (c) rewriting the Google code to use new software dependencies from the target environment. Moreover, even with a great deal of upfront planning, it would be impossible for SWEs to know in advance the full universe of the software code that would need to be rewritten, which would likely lead to delays as the engineering project unfolds. In other words, good planning is essential, but realistically can never cover every eventuality.

167. Another approach the SWEs may take, after assessing the code that needs to be separated and divested, may be to rewrite from scratch without "lifting" any preexisting code. Depending on the identity of the buyer and the scoping work done to create an architectural plan, a complete rewrite may ultimately be more efficient than "lift-and-shift."[395] But a rewrite from scratch would, of course, involve its own complexities due to the difficulty of reconstructing such a complex system with at least tens of millions of lines of code without including all of its dependencies.[396]

2. Establishing a Hardware Infrastructure Outside Google for the Divestiture Product(s)

168. As part of any divestiture, a buyer would have to provision hardware computing, storage, and networking capacity on the infrastructure on which it plans to operate the to-be-created AdX or DFP. This would not be straightforward to accomplish given the hardware infrastructure that would be required, both in terms of scale[397] and geographic distribution,

[395] Nitish Korula Deposition, 6/9/25 at 151:3-153:3 ("So the reason that I think this is hard is because you mentioned a couple times, separated out from the current stack. I actually wouldn't do that, right. In the sense that it's so integrated into Google systems, that depend on other Google systems, that depend on other Google systems, that what I would do is start from scratch, if you will. I would say, okay, you want to build a system that replicates much of this functionality. And instead of trying to separate the existing system, you would say, okay, I'm going to build a new system that, as best as I can, duplicates the old system ..."); at 179:10-180:5 ("On the other hand, it is sometimes the case that systems have so much complexity that trying to work in a system of considerable complexity is actually inherently slow. And it would be faster to do something from scratch.").

[396] For example, a single directory in the recent GAM code snapshot produced by Google as part of remedies phase discovery in this Action (/google3/contentads) contains 19,694,124 non-blank, non-comment lines of code as measured by cloc-2.06. And that directory is only a portion of the GAM code produced by Google in this action.

[397] Even existing ad tech providers would need to add significant capacity to accommodate the addition of AdX or DFP. *E.g.*, Arnaud Creput Deposition (Equativ), 6/11/25 at 71:13-19 ("[redacted]).

to meet necessary latency, reliability, and performance requirements (*see* Section II.C).[398] Furthermore, provisioning such infrastructure requires extra time to plan (i.e., logistics and operations, separate from the architecture development plan), raise capital, speak with vendors and providers, and also hire additional engineers as needed.[399] In light of these requirements and restrictions, a divestiture buyer could consider several alternatives.

169. One option would be for a buyer to build its own hardware infrastructure on which to operate AdX or DFP from scratch. This might entail building or leasing data center space, as well as provisioning the required power and networking infrastructure.[400] Existing hyperscale infrastructure providers have typically found it efficient to build and maintain their own infrastructure.[401]

[398] George Levitte, 6/13/25 at 91:1-92:5 [redacted]).

[399] Heather Adkins Deposition, 6/12/25 at 65:15-23 (Google core engineer testifying that "I would say hardware upgrades are some of the longest deployments that we have. Hardware is expensive. It takes time to physically carry things to machines and swap them out.").

[400] Betsy Beyer et al., "Site Reliability Engineering: How Google Runs Production Systems - The Evolution of Automation at Google" *O'Reilly* (2016) https://sre.google/sre-book/automation-at-google/ (Accessed July 7 2025) ("The steps taken to get a cluster ready for use were something like the following: 1. Fit out a datacenter building for power and cooling. 2. Install and configure core switches and connections to the backbone. 3. Install a few initial racks of servers. 4. Configure basic services such as DNS and installers, then configure a lock service, storage, and computing. 5. Deploy the remaining racks of machines. 6. Assign user-facing services resources, so their teams can set up the services. Steps 4 and 6 were extremely complex. While basic services like DNS are relatively simple, the storage and compute subsystems at that time were still in heavy development, so new flags, components, and optimizations were added weekly."). Note that this is a simplification of the process, and outlines the initial setup steps for a new cluster—not an entire data center at Google—and is intended as an overview for new Site Reliability Engineering hires at Google.

[401] Luiz André Barroso et al., *The Datacenter as a Computer: Designing Warehouse-Sale Machines, Third Edition*, Morgan & Claypool Publishers (2018) at 38 ("Large-scale internet service providers such as Google usually take a different approach, in which both application-specific logic and much of the cluster-level infrastructure software is written in-house. Platform-level software does make use of third-party components, but these tend to be open-source code that can be modified in-house as needed. As a result, more of the entire software stack is under the control of the service developer. … This approach adds significant software development and maintenance work but can provide important benefits in flexibility and cost efficiency. Flexibility is important when critical functionality or performance bugs must be addressed, allowing a quick turnaround time for bug fixes at all levels. It eases complex system problems because it provides several options for addressing them. For example, an unwanted networking behavior might be difficult to address at the application level but relatively simple to solve at the RPC library level, or the other way around.").

170. As a reference point, Google generally opts to use or build its own infrastructure for its internal projects because Google-built data centers are highly efficient and optimized,[402,403] in contrast to smaller competitors that may rely more heavily on rental agreements with third-party data centers[404] or public cloud providers such as AWS. [redacted].[405] [redacted] (*see* Section IV.B). Building a data center requires numerous phases that in brief consist of: planning and obtaining funding, selecting a site (i.e., land and physical space), contracting with power suppliers, constructing the data center facilities, and establishing and activating high-speed networking infrastructure.[406] Construction of data centers may also be subject to any delays imposed by regulations in the relevant countries.[407] Figure 13 depicts the complexity of and additional steps required for what Google refers to as a "turnup" of a data center.

[402] GOOG-AT-MDL-B-009837031, at -040 ("YAWN - Yet Another Willpower Name … YAWN data centers achieve efficiencies in cost, energy, and construction speed by being optimized specifically for Google-designed servers cooled by Google-designed cooling systems, and powered by a Google-designed power architecture. … The name 'YAWN' is an acronym that stands for 'Yet Another Willpower Name,' a reference to the 2004 Willpower project that was (essentially) Google's first data center.").

[403] GOOG-AT-MDL-B-009836306, at -309 ("A YAWN (or 'YAWN site') is a datacenter building that is designed, built and owned by Google and that has been specially configured to use custom-built racks, cooling, and other Google-specific infrastructure. YAWN is an initialism for 'Yet Another Willpower Name'. Willpower was the name of Will Whitted's project to create Google- designed and built datacenters. YAWNs are regularly staffed by Google employees.").

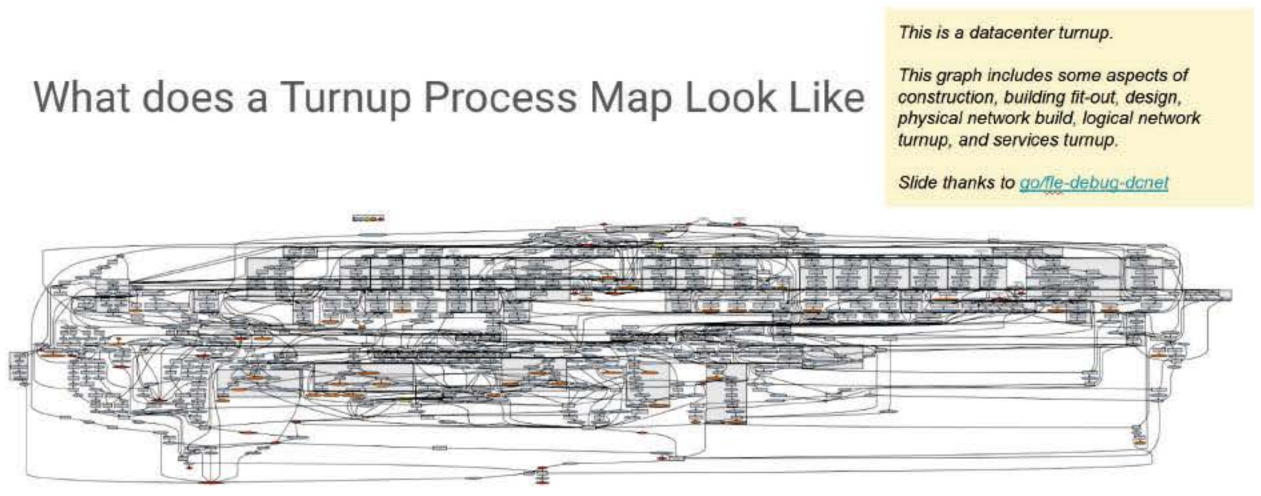
[404] Andrew Casale Deposition (Index Exchange), 7/2/25 at 110:23-111:7 ("[redacted]").

[405] GOOG-AT-MDL-B-009836980, at -982 ("[redacted]").

[406] GOOG-AT-MDL-B-009837031, at -042 ("Lifecycle … PA Demand … Funding … Construction & Network … Machines …").

[407] Data centers must comply with numerous legal regulations, certifications, and standards. These requirements can vary significantly not only by country, but even at the state level (i.e. for the US) or even city level. Some examples of such requirements include data privacy laws such as the GDPR in the EU; the Health Insurance Portability and Accounting Act ("HIPAA") and the Children's Online Privacy Protection Rule ("COPPA") at the federal level in the US, and the California Consumer Privacy Act ("CCPA") and the New York Stop Hacks and Improve Electronic Data Security ("SHIELD") Act at the state level in the U.S. Data centers also generally adhere to various quality, security, and environmental standards defined by various international standards organizations such as the International Organization for Standardization ("ISO").

Figure 13: Google YAWN Data Center "Turnup" Process Map[408]

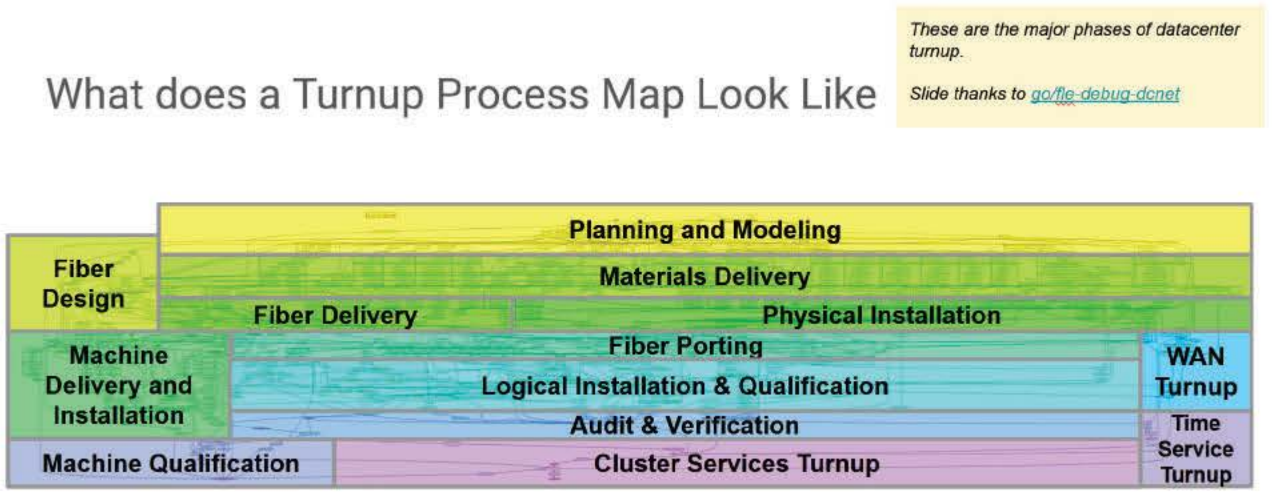


What does a Turnup Process Map Look Like

This is a datacenter turnup.

This graph includes some aspects of construction, building fit-out, design, physical network build, logical network turnup, and services turnup.

Slide thanks to go/fle-debug-dcnet



What does a Turnup Process Map Look Like

These are the major phases of datacenter turnup.

Slide thanks to go/fle-debug-dcnet

171. As a result, the lead times for building sufficient hardware infrastructure for AdX or DFP would take many years. This would especially be the case given that the latency, reliability, and performance requirements for running AdX or DFP would necessitate that the hardware infrastructure be located in multiple geographic regions.[409] Moreover, constructing new hardware infrastructure would come with the added challenge that the new hardware infrastructure would not come with pre-existing software infrastructure running on the hardware infrastructure, unless the buyer were a data center provider with its own high performance software infrastructure.

[408] GOOG-AT-MDL-B-009837031, at -072 and -074.
[409] *See* Section II.C.

172. Another option might be for a buyer to expand its existing hardware infrastructure to accommodate AdX or DFP. But given the scale at which AdX and DFP operate, the feasibility of this option would depend on the capabilities of the buyer's existing hardware infrastructure and whether those capabilities could even be expanded to the scale necessary for DFP or AdX.[410] For example, even an ad tech provider with some existing infrastructure like Index Exchange would likely need to more than double, or even triple, the size of its entire current global infrastructure to match the computing resources that AdX functionality alone utilizes—without even including all the shared services on which AdX relies.[411] (If shared Google services utilized by GAM are included, then Index Exchange would need to increase the size of its global infrastructure by more than 20 times.[412]) Accordingly, this option would likely only be a realistic option for buyers who already have existing hyperscale infrastructure[413] for the same reasons building entirely new infrastructure, as discussed above, would be challenging.

173. A third option for a buyer would be to use a public cloud computing platform, such as AWS, Azure, or GCP. The advantage of this option is that it would allow the buyer to leverage existing hardware infrastructure offered by the cloud computing platform and also delegate to the public cloud provider *some* aspects of the problem of scaling up the infrastructure to AdX's or DFP's needs. Even so, simply relying on existing cloud infrastructure for hardware would only take care of one part of the steps required to divest. Google and the buyer would still need to complete the significant work required to make AdX or DFP work with new software infrastructure. In addition, the computing platform would need to offer sufficient software infrastructure support to meet the buyer's needs or the buyer would still need to build necessary software dependencies (*see* Section V.A.3 below).[414] In addition, the

[410] For example, [redacted] *See* James Avery Deposition (Kevel), 6/24/2025 at 23:22-26:11. [redacted]

[411] [redacted] *See* IX00000569, at -577 ("[redacted]"). AdX alone runs on 990 K GCUs, which is roughly around 500,000 to 800,000 physical CPU cores. *See* Google's Responses to Plaintiffs' Third Set of Interrogatories at 8 (describing 2025 GCU usage for GAM).

[412] Assuming [redacted], and assuming 12.3 M GCUs for shared Google services to support GAM. *See* IX00000569, at -577. ("[redacted]"); Google's Responses to Plaintiffs' Third Set of Interrogatories at 8 (describing 2025 GCU usage for GAM).

[413] *See* Amazon, Microsoft, Meta (Facebook), and Apple as illustrative examples.

[414] Cloudflare, Inc, "What is a public cloud? | Public vs. private cloud," *Cloudflare Learning Center*, https://www.cloudflare.com/learning/cloud/what-is-a-public-cloud/ (Accessed July 6, 2025) ("What are the pros and cons of using a public cloud? … Pros: … Less server management: If an organization uses a public cloud, internal teams don't have to spend time managing servers – as they do for legacy on-premises data centers or for internal private clouds.").

downside of passing off responsibility for the hardware infrastructure to a cloud provider is that this reliance will very likely incur higher costs[415,416] and result in less control over the infrastructure,[417,418,419] making it more difficult to ensure AdX and DFP continue to perform at the high level required by customers.

174. If a buyer were to build new data centers from scratch or expand its existing data centers, it would need to decide whether to buy or rent servers. Even if a buyer were to use a cloud computing platform, it is likely that a buyer would have to negotiate with the provider over the installed hardware that AdX or DFP would use. For the scale, speed, and high performance at which AdX and DFP operate, it is likely that even a hyperscale cloud computing platform such as AWS or Azure would have to provision new hardware specifically for the buyer, as well as potentially new facilities, if they could support AdX or DFP at all,[420] both of which would involve considerable time spent in negotiation and

[415] Relying on public cloud infrastructure replaces "fixed costs" with "variable costs" that grow with the amount of services required. *See* Stephanie Layser Deposition (ex-News Corp, current Amazon), 7/1/2025 at 178:14-179:8; 179:22-180:20 ("[redacted]).

[416] [redacted] GOOG-AT-MDL-018760130, at -135. Another product anticipated saving $480 million across five fiscal years, with a reduction in average monthly cost of 47%, just from transitioning from GCP to Google internal infrastructure. GOOG-AT-MDL-004123401, at -423 ("[M]ajor cost savings from GCP to Borg transition ($480M FY21-FY25 savings"); -441 ("[M]argin improvements from materially lower cloud costs resulting from true EBR and shift from GCP to TI"); -447 ("Shifting from GCP to TI/Borg brings down avg. monthly cost by ~47%").

[417] Luiz André Barroso et al., *The Datacenter as a Computer: Designing Warehouse-Sale Machines, Third Edition*, Morgan & Claypool Publishers (2018), at 38 ("Large-scale internet service providers such as Google usually take a different approach, in which both application-specific logic and much of the cluster-level infrastructure software is written in-house. … As a result, more of the entire software stack is under the control of the service developer. This approach adds significant software development and maintenance work but can provide important benefits in flexibility and cost efficiency. Flexibility is important when critical functionality or performance bugs must be addressed, allowing a quick turnaround time for bug fixes at all levels. It eases complex system problems because it provides several options for addressing them.").

[418] George Levitte Deposition, 6/13/25 at 60:13-62:19 ([redacted]").

[419] GOOG-AT-MDL-018760130, at -134 ("[redacted]").

[420] *See* Brad Smith, "Microsoft-LinkedIn deal cleared by regulators, opening new doors for people around the world," Official Microsoft Blog (Dec. 6, 2016) https://blogs.microsoft.com/blog/2016/12/06/microsoft-linkedin-deal-cleared-regulators-opening-doors-people-around-world/(discussing Microsoft's acquisition of LinkedIn). *See also* Jordan Novet, "LinkedIn shelved planned move to Microsoft Azure, opting to keep physical data centers,"

additional lead time. Additional considerations would factor into these decisions, such as whether the buyer would seek to use "bare metal" servers (i.e., its own dedicated servers) or virtualized instances of servers (i.e., servers that are shared among potentially multiple different cloud customers).

175. Portions of Google's ad tech systems, [redacted],[421] also depend on TPUs, which are Google-developed, customized hardware designed for AI and ML workloads.[422] The relative difficulty of procuring these custom-built TPUs, which are a near necessity for performing AI workloads efficiently at large scale,[423] would add to the challenge of establishing a hardware infrastructure to fully replicate the functionality and features GAM has today.

3. Establishing a Software Infrastructure Outside Google for the Divestiture Product and Writing New Versions of the Product That Work in the Buyer's Software Infrastructure

176. A significant challenge to divestiture is that it is not possible to simply move the existing software providing ad serving or exchange functionality to data centers owned by a divestiture buyer. As discussed extensively above in Sections III and IV, GAM is highly dependent on and has been developed in tandem with custom, proprietary Google technology that is not compatible with non-Google infrastructure.

CNBC (Dec. 14, 2023) https://www.cnbc.com/2023/12/14/linkedin-shelved-plan-to-migrate-to-microsoft-azure-cloud.html (discussing how LinkedIn set aside its effort to relocate its data center technology into Microsoft's Azure Cloud, three years after the announced migration).

[421] *See e.g.*, GOOG-AT-MDL-B-007171881 ([redacted]).

[422] Tensor Processing Units ("TPUs") are custom hardware developed by Google that are optimized for computations related to AI/ML. *See* "Accelerate AI development with Google Cloud TPUs," *Google Cloud* https://cloud.google.com/tpu?hl=en, (Accessed July 6, 2025) ("Google Cloud TPUs are custom-designed AI accelerators, which are optimized for training and inference of AI models.").

[423] https://cset.georgetown.edu/publication/ai-chips-what-they-are-and-why-they-matter/, (Accessed July 7, 2025). ("This enormous computational power is delivered by computer chips that … are tailor-made to efficiently perform specific calculations required by AI systems. Such leading-edge, specialized 'AI chips' are essential for cost-effectively implementing AI at scale; trying to deliver the same AI application using older AI chips or general-purpose chips can cost tens to thousands of times more. The fact that the complex supply chains needed to produce leading-edge AI chips are concentrated in the United States and a small number of allied democracies provides an opportunity for export control policies.")

177. Because GAM has significant dependencies on Google's proprietary software infrastructure and Google's proprietary infrastructure would not be available to a buyer,[424,425] a buyer must create or already have a substitute for these software infrastructure dependencies, and the divested product must work with the buyer's version of the dependencies.[426] Without an adequate substitute for these dependencies, the divested product would not work. Even if the divestiture buyer were able to provide its own hardware infrastructure (data centers), the SWEs would need to either already have or replace Google's custom dependencies, such as Borg, Colossus, Spanner, F1, and many others, which all form the backbone of Google's software infrastructure.

178. Moreover, for a divestiture buyer to be able to operate AdX or DFP, SWEs must design and write new versions of AdX and DFP that can work outside Google's infrastructure and in the buyer's. As explained below, implementing both of these tasks while minimizing disruption to customers would be an enormous undertaking.

[424] This would remain true even if a buyer were to use GCP because, as noted above, the infrastructure for Google's proprietary software environment (G3) vs. Google's public cloud service (GCP) is substantially different. *See generally* GOOG-AT-MDL-018760130; *supra* section III.

[425] Other ad tech companies vary in the software infrastructure that their own tools depend on. Some rely on software provided by third parties, others their own proprietary software or a combination. Arnaud Creput Deposition (Equativ), 6/11/2025 at 111:17-112:4 ("[redacted]; James Avery Deposition (Kevel), 6/24/2025 at 31:25-33:1 ("[redacted]); 35:7-15 ([redacted]") (Note that [redacted] has no relation with Google's internal and bespoke Dynamo-as-a-Service).

[426] *E.g.*, Arnaud Creput Deposition (Equativ), 6/11/2025 at 120:5-121:12 ("[redacted]").

179. To achieve such a software migration, SWEs might consider a combination of at least three options—all of which would require writing substantial amounts of new code:

a. *Rewriting AdX or DFP dependencies.* This approach would involve SWEs first identifying the proprietary software infrastructure code on which GAM's ad serving and exchange functionality (and many other Google products) relies and then creating a new version that lives in the buyer's infrastructure and that the divested product can use. For example, GAM's ad serving and exchange functionality rely on Google's proprietary Borg, Colossus, and Sawmill systems (among many other proprietary software systems), so a buyer would have to create a replacement for each of these software systems. To the extent SWEs could recreate these software dependencies to be near replicas of Google's software infrastructure, the code for GAM's ad serving or exchange functionality could then theoretically be reconnected to the new dependencies with minimal code rewriting—assuming the ad serving and exchange functionalities can be disentangled from one another in the first place.

i. Given the size and complexity of the dependencies, this approach would be an enormous undertaking and would require a team of SWEs to rewrite millions of lines of code and therefore recreate many years of research and code development time. For example, Google's researchers and engineers spent over 5 years developing and iterating on Spanner before publishing their first research paper describing it.[427] SWEs would need to replicate much more than just those 5 years of work, given ongoing Spanner improvements and that 10 years have elapsed since Spanner's original publication. And that is only one example of many dependencies.

ii. Attempting to recreate all the dependencies from scratch would come with the added complexity of not being able to test each component with other dependencies that are known to work. This would be more challenging compared to replacing portions of a known-working system. For example, a SWE may be rewriting a GAM service that originally depended on the AdSpam infrastructure for detecting invalid traffic. But given that the divested GAM functionality is being rewritten to work outside of Google, and given that the AdSpam team is separate from the GAM engineering team, the divestiture SWE may have no access to AdSpam systems and thus be unable to rely on the AdSpam infrastructure while writing code that depends on it. The SWE would essentially need to start from scratch

[427] James C. Corbett et al., Spanner: Google's Globally Distributed Database, *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation* (OSDI 2012) at 1-22 ("Since Spanner's inception, we have taken more than 5 years to iterate to the current design and implementation.").

without any dependencies for invalid traffic detection that are known to work.

iii. There is also the question of the process for writing the new dependencies. Because the new system would be hosted outside of Google's ecosystem in the intended target environment, e.g., hypothetically, on a public cloud provider such as AWS or on the internal hyperscale infrastructure of another company, Google might have to construct an interface to its proprietary systems that could be accessed from the outside in order to enable data transfer between the two environments and allow engineers to reference existing Google systems for building new ones. Doing so would be a substantial and separate engineering project in its own right, create significant security risks for Google, and expose decades of Google's proprietary technological development that were designed to support many other Google products. Moreover, creating an interface would be a gating first step and could not occur concurrently to SWE testing of new infrastructure components with old working ones.

b. *Writing a shim*.[428] Another approach SWEs may use is to write a "shim" layer of code that translates between DFP's software and a new set of dependency software.[429] This approach is, however, limited in at least two ways: (1) the more differences there are in the divestiture buyer's software environment compared to Google's, the more difficult it will be to design and write shim code. If—as is likely—the buyer's environment differs enough from Google's environment, the shim strategy may collapse into the labor-intensive approach, described above, of rewriting Google's existing dependencies. (2) The SWEs writing the shim layer would greatly benefit from being domain experts in *both* the new environment and Google's current dependencies, and it is likely that very few—if any—such SWEs exist (and certainly not the hundreds I assume would work on divestiture). A shim can only correctly and efficiently translate between the divestiture product's current code dependencies and the dependencies in the new environment if the engineer who wrote it is extremely knowledgeable in both sides of the shim.

i. Another typical challenge with this approach is identifying whether sufficiently equivalent technology features exist in the new dependencies to which the shim layer directs the code. For example, although Kubernetes

[428] A shim layer is a code that effectively translates between two incompatible components or layers of a software system to provide compatibility.

[429] "Shim," *mdn web docs*_, https://developer.mozilla.org/en-US/docs/Glossary/Shim (Accessed June 24, 2025) ("A shim is a piece of code used to correct the behavior of code that already exists, usually by adding new API that works around the problem.").

is open source and is also a cluster manager like Borg,[430] the two systems have different designs and are not compatible with each other, and Kubernetes does not even support the same scale as Borg.[431] Borg also provides much finer-grain control for individual software and access permissions, which are both crucial for scheduling tasks running directly on bare metal servers. That is in contrast to Kubernetes, which uses a completely different containerized approach based on isolation.[432] Even if there ostensibly may be analogues to other custom Google technologies, SWEs would have to scrutinize the analogues to ensure that they are truly equivalent in terms of features, performance, and efficiency or, if not, consider what other changes to AdX or DFP might be necessary to compensate for or adapt to the differences in supporting infrastructure.

c. *Rewriting code to use new dependencies*. If writing a shim layer is not feasible because the environments differ, SWEs might also rewrite the code supporting GAM's ad serving and exchange functionality to rely on entirely new dependencies. But, similar to the shim approach, this approach only works if the buyer's dependencies are already substantially equivalent to the Google dependencies in terms of features, performance, and efficiency. Moreover, it would be challenging to execute because SWEs with familiarity in Google's ad tech would need to rewrite voluminous code that Google developed incrementally over more than 10 years.

[430] Both Borg and Kubernetes are cluster managers developed by Google. Although they share some design principles, they are distinct enough that Borg continues to be Google's internal-use cluster manager, while Kubernetes is offered as a commercial service. *See* Abhishek Verma et al., "Large-Scale Cluster Management at Google with Borg," *Proceedings of the Tenth* European *Conference on Computer Systems (EuroSys)*, ACM (April 2015), at 13 ("Google's open-source Kubernetes system [53] places applications in Docker containers [28] onto multiple host nodes. It runs both on bare metal (like Borg) and on various cloud hosting providers, such as Google Compute Engine. It is under active development by many of the same engineers who built Borg….We discuss how lessons from Borg are being applied to Kubernetes in the next section.").

[431] Kubernetes states that it is specifically designed for clusters that scale only up to certain limits. *See* "Considerations for large clusters," *kubernetes* (last modified April 26, 2024) https://kubernetes.io/docs/setup/best-practices/cluster-large/ (Accessed July 6, 2025) ("More specifically, Kubernetes is designed to accommodate configurations that meet all of the following criteria: … No more than 110 pods per node … No more than 5,000 nodes … No more than 150,000 total pods … No more than 300,000 total containers."). In contrast, I am not aware of any such similar practical limitations on scale imposed by Borg on Google systems. *See* Verma et al., "Large-Scale Cluster Management at Google with Borg," *Proceedings of the Tenth* European *Conference on Computer Systems (EuroSys)*, ACM (April 2015) at 5 ("We are not sure where the ultimate scalability limit to Borg's centralized architecture will come from; so far, every time we have approached a limit, we've managed to eliminate it.").

[432] Glenn Berntson Deposition, 6/24/25 at 167:15-171:14 ("Borg was designed to run off of bare metal servers and very fine-grain granularities of control for individual binaries and the way permissions are managed across all the RPC boundaries, at a level in granularity that is a complete mismatch to the model of all modern cloud computing, which is all containerized, where you've got sort of these snapshots that you load into binaries. They are completely different models for how you build, manage, deploy, and run systems.").

180. Given the complexity of Google's software infrastructure, it would be unrealistic for a buyer to pursue a strategy of rewriting all of Google's dependencies on target infrastructure in any reasonable timeframe. Instead, the most likely strategy for a buyer would be some combination of the three options—even though each comes with its own challenges.

181. A buyer would likely first look for potential software substitutes that could serve as a primitive basis for replacing aspects of Google's software infrastructure and then build additional necessary software infrastructure on top of these substitutes to attempt to provide, to whatever extent possible, an operating environment that retains some degree of compatibility with Google's software infrastructure on the target infrastructure.

182. SWEs would then need to write entirely new code to enable AdX or DFP to operate in the new operating environment on the target infrastructure. At a minimum, this would require writing a shim layer to translate all previous function calls (i.e., calls to particular services offered by supporting code) to previous Google infrastructure frameworks, and all protocol messages (i.e., communications between software systems) for interacting with previous Google infrastructure services, to communicate with the frameworks and services available in the new operating environment. This would likely require building new frameworks and services to bridge any likely differences between what the new operating environment provides to AdX or DFP and what was provided by the previous Google infrastructure. Beyond writing a shim layer, I would expect that there would also be substantial changes required to the divestiture product's code itself due to differences between the previous Google infrastructure and the new operating environment.[433]

183. For example, Google's ad tech and other custom software rely on tens of thousands of BCL files to run on Google infrastructure using Borg.[434] In a divestiture, these tens of thousands of Borg configuration files would need to be converted to a form usable by AdX or DFP outside of Google infrastructure. Depending on the circumstances of the divestiture buyer, some combination of the three options described above (rewriting, shimming, or replacing) would be needed for all the BCL files. The likely case is that the buyer does not have a Borg equivalent in terms of its performance and scalability. It would then be necessary to first create a Borg replica to match its performance and scalability requirements, which itself would require significant implementation effort and extensive testing. In the unlikely case

[433] George Levitte Deposition, 6/13/25 at 102:9-20 ("[redacted].").

[434] Abhishek Verma et al., "Large-Scale Cluster Management at Google with Borg," *Proceedings of the Tenth European Conference on Computer Systems (EuroSys)*, ACM (April 2015) at 2 ("[T]ens of thousands of BCL files are over 1 k lines long, and we have accumulated tens of millions of lines of BCL.").

that the divestiture buyer has its own Borg equivalent,[435] then it may be more efficient to either write a shim to translate from the Borg BCL files to the divestiture buyer's Borg-equivalent system, or replace the dependence on Borg entirely.

184. Another part of Google's proprietary infrastructure that leverages configuration files is its experiment infrastructure, Mendel, which depends on thousands of GCL files that configure thousands of Mendel experiments.[436] Google SWEs depend on Mendel for running experiments,[437] and on RASTA for analyzing and reporting the data from those experiments.[438] Demonstrating how unique to Google's internal infrastructure these dependencies are, Mendel experiments are typically configured using (Mendel) GCL files,[439] which in itself is a bespoke configuration language developed by Google. [redacted].[440] This includes instructions for running the experiment such as telling Mendel which part of Google's software should be run for the experiment, how the various variables, flags, and options for the software should be modified for the experiment, and sometimes what infrastructure configuration to use. GCL files are similar to Borg configuration files in that they can both be used to configure parameters for Google's software.[441] These experiments are therefore all conditionally enabled, or active, in Google's production software, and to exist outside Google's infrastructure would need to be converted to a new form with similar caveats as with Borg BCL files. As with Borg, it is also unlikely that any divestiture buyer has a Mendel equivalent, as well as an equivalent to RASTA for analyzing Mendel data. Any approach used to replicate Mendel and RASTA's functionalities would therefore require time and effort to develop.

[435] To the best of my knowledge, there are no off-the-shelf alternatives that match Borg's performance and scalability. Meta's internal proprietary cluster manager named Twine is one of the few contenders. *See generally* Chunqiang Tang, et al., "Twine: A unified cluster management system for shared infrastructure," *14th USENIX Symposium on Operating Systems Design and Implementation* (Nov. 2020), https://www.usenix.org/conference/osdi20/presentation/tang.

[436] GOOG-AT-MDL-B-009834598, at -598 ("Here are some quick stats (from 2024/11/06) for Display Ads experiments: … • ~175 average experiments selected per request … • ~13000 total active experiments … • ~800 new experiments started every week.").

[437] In November 2024, there were approximately 13,000 experiments. *See* GOOG-AT-MDL-B-009834598, at -598 ("Here are some quick stats (from 2024/11/06) for Display Ads experiments: … • ~175 average experiments selected per request … • ~13000 total active experiments … • ~800 new experiments started every week.").

[438] Jason Hsueh Deposition, 11/15/23, at 18:10-19:1 ("A. … RASTA has a specific purpose of evaluating metrics for changes within the ad serving system. … Q. … So is RASTA used to evaluate the metrics for all changes within the ad serving system? … A. Anything that's run as an experiment will typically go through RASTA.").

[439] GOOG-AT-MDL-B-009836267, at -267 ("Create & Ramp … Mendel GCL … Use Mendel Configuration Language to code and manage your studies and launches …").

[440] GOOG-AT-MDL-018528378, at -395-396. ("[redacted]").

[441] For exam le, *see generally* [redacted] ("[redacted]").

185. As another example, AdX's and DFP's software depend on F1[442] for their operation and include various components that depend on the specific operational characteristics of F1. Those components would likely need to be modified, as the database system provided in the target infrastructure would likely have different operational characteristics than F1. For example, F1 database table rows support direct use of custom Google data structures,[443] which is typically not a supported feature in other relational databases.[444] A divested AdX or DFP would therefore need to work without F1's support for Google's data structures, either by writing new code to support Google's data structure or choosing a new data format that matches their performance.

186. The above examples are just a few of the many dependencies that AdX and DFP rely on in Google's bespoke proprietary infrastructure. As I explained above, even each one standing alone would require identifying, obtaining, or creating an equivalent to Google's infrastructure, and writing code to enable AdX and DFP to work in the new infrastructure.[445,446,447]

[442] GOOG-DOJ-13940968, at -968-69 ("Agave is an infrastructure effort started on DFP … I extended the scope to include AdX data. I defined the strategy for and executed a multi-year large scale migration of AdX systems to F1/Spanner …"). *See also* GOOG-AT-MDL-B-009836576, at -576 ("AdX publisher data currently resides in MySql. There is an effort across ads to move off of MySql to F1. As part of that, AdX publisher data will also be migrated from MySql to the DRX Agave F1 instance.").

[443] Protocol Buffers ("Protobufs") are a Google-developed data format that is open and free, and is extremely efficient and high performance.

[444] F1 databases support the direct use of Google's Protobuf data format in table rows. Typically, relational databases do not support the use of complex data structures within data rows, i.e., complex queries cannot be meaningfully performed on them. *See* Jeff Shute, et al., "F1: A Distributed Database that Scales," *Proceedings of the VLDB Endowment*, Volume 6, Issue 11 (August 2013) at 3 ("The F1 data model supports table columns that contain structured data types. These structured types use the schema and binary encoding format provided by Google's open source Protocol Buffer [16] library. Protocol Buffers have typed fields that can be required, optional, or repeated; fields can also be nested Protocol Buffers. At Google, Protocol Buffers are ubiquitous for data storage and interchange between applications. When we still had a MySQL schema, users often had to write tedious and error-prone transformations between database rows and in-memory data structures. Putting protocol buffers in the schema removes this impedance mismatch and gives users a universal data structure they can use both in the database and in application code.").

[445] George Levitte Deposition, 6/13/25 at 102:9-20 ("[redacted]).

[446] Nitish Korula Deposition, 6/9/25 at 241:6-18 ("Almost every aspect of Google's infrastructure is unique. Many innovations in large-scale distribut[ed] systems design came from Google. Many of them were subsequently published, but Google's internal portions are different from the public-released portions. Everything from the databases, to the final storage systems, to the machine learning systems, to the login systems, to the load balancing systems, just really all of it is unique to Google.").

[447] Betsy Beyer et al., "Site Reliability Engineering: How Google Runs Production Systems - The Evolution of Automation at Google" *O'Reilly* (2016), https://sre.google/sre-book/automation-at-google/, (Accessed July 6, 2025) ("We [Google] have built APIs for systems when no API was available from the vendor. Even though purchasing software for a particular task would have been much cheaper in the short term, we chose to write our own solutions, because doing so produced APIs with the potential for much greater long-term benefits.").

187. Other software engineering best practices would further complicate the migration process. For example, engineers would need to establish an appropriate testing and deployment architecture. Today, GAM relies on proprietary Google systems for such tasks, but those systems would not be available in (or tailored to) the target environment.[448] SWEs would therefore need to create a plan for not only deploying AdX or DFP for the very first time, but also for reliably deploying it every time its code is updated, while ensuring that AdX or DFP experiences no or minimal disruptions to service. For example, it is common for SWEs to architect "continuous integration/continuous deployment" ("**CI/CD**") pipelines built for the sole purpose of automatically, rigorously, and continuously testing code before deploying it to the underlying infrastructure.[449] CI/CD pipelines consist of custom software dependencies specifically made for testing and deployment, and therefore require a team of SWEs dedicated to maintaining the CI/CD pipelines and adding features. Since AdX's and DFP's software would very likely be entirely new software that runs in a completely different infrastructure from Google's ad tech, AdX's and DFP's CI/CD pipelines will also likely need to be completely re-designed.

4. Enabling Scalability

188. Beyond just establishing a working version of AdX's or DFP's software that runs correctly within the new software environment and contains all of the necessary functionality of the Google version, SWEs will need to enable the divested product to scale to meet the demands of the customer base. It is one thing to get a version of AdX's or DFP's software up and running on the new infrastructure (itself a major challenge, as discussed above), but it is another thing to ensure that the divested software can actually satisfy the demands put on it

[448] Software deployment is a complicated enough process that Google dedicates specific job titles to its associated responsibilities. *See* Betsy Beyer et al., *Site Reliability Engineering: How Google Runs Production Systems - Service Level Objective,* O'Reilly (2016, https://sre.google/sre-book/release-engineering/, (Accessed July 6, 2025) ("Release engineering is a relatively new and fast-growing discipline of software engineering that can be concisely described as building and delivering software [McN14a]. Release engineers have a solid (if not expert) understanding of source code management, compilers, build configuration languages, automated build tools, package managers, and installers. Their skill set includes deep knowledge of multiple domains: development, configuration management, test integration, system administration, and customer support. … Release engineering is a specific job function at Google. Release engineers work with software engineers (SWEs) in product development and SREs to define all the steps required to release software—from how the software is stored in the source code repository, to build rules for compilation, to how testing, packaging, and deployment are conducted.").

[449] Red Hat, Inc., "What is CI/CD?" *Red Hat* (Dec. 12, 2023) https://www.redhat.com/en/topics/devops/what-is-ci-cd (Accessed July 6, 2024) ("Continuous integration (CI) refers to the practice of automatically and frequently integrating code changes into a shared source code repository. Continuous delivery and/or deployment (CD) is a 2 part process that refers to the integration, testing, and delivery of code changes. Continuous delivery stops short of automatic production deployment, while continuous deployment automatically releases the updates into the production environment.").

by the customer and user base of AdX or DFP at the transaction volumes it operates at today.[450]

189. Scaling introduces entirely new problems to address, as evidenced by the years,[451] and in some cases, decades,[452] of research and engineering that Google dedicated to scaling its custom and proprietary systems such as Colossus and Spanner so that those systems could satisfy the particular needs of Google's products. For example, when Google acquired DoubleClick, it completely rewrote DoubleClick ad serving and exchange code and systems to be entirely based on Google's infrastructure in a process that took almost 7 years.[453] And even that project was much simpler than the divestiture of AdX or DFP because Google SWEs could leverage years of existing Google research spent on developing custom infrastructure that scales. Furthermore, the original DFP that was migrated from DoubleClick has grown together with Google as a whole in the approximately 10 years since the completion of the DoubleClick acquisition. For example, the ad serving functionality in GAM currently handles 65 times more queries per second than DFP did in

[450] Part of the scaling up of software to meet customer demands involves not only scaling infrastructure, but business operations in general which require in-house tooling and support software. Scaling up a system thus is significant enough that it is often treated as an entirely separate step in the software engineering process, such as in the case of Google Display Network ("GDN"), formerly known as Google Content Network ("GCN"). In 2009, Google implemented and delivered many GDN features to their customers, and as a result this early version of GDN rapidly became the "leading ad network" of the time. Despite its widespread success, Google believed that GDN lacked the capability to scale globally, and therefore one of Google's goals for the year 2010 was to scale GDN. *See* GOOG-TEX-00825713, at -716 ("Scale GCN! Now that we have delivered scores of features making GCN the leading ad network, we need to scale our business operations. We must build internal tools to enable our sales force to sell and service our advertiser and publisher partners around the world.").

[451] Google's research paper describing Spanner was published in 2012, and represented about 5 years of engineering effort by then. This suggests that Spanner, at this time of writing, has undergone approximately 17 years of development and usage at Google. *See* James C. Corbett et al., "Spanner: Google's Globally Distributed Database," *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation* (OSDI 2012) at 8:20 ("Since Spanner's inception, we have taken more than 5 years to iterate to the current design and implementation.").

[452] At the time of writing, Google's research paper describing Borg was written and published nearly 10 years ago in early 2015, and had been in use for a decade already, suggesting that Borg has seen nearly two decades of development and usage. *See* Abhishek Verma et al., "Large-Scale Cluster Management at Google with Borg," *Proceedings of the Tenth* European *Conference on Computer Systems (EuroSys)*, ACM (April 2015) at 14 ("Virtually all of Google's cluster workloads have switched to use Borg over the past decade. We continue to evolve it…").

[453] Approximately 100 SWEs spent nearly 7 years total to rewrite DoubleClick and move it to Google infrastructure (50 SWEs were tasked with the rewrite, and 50 SWEs maintained and improved the old DoubleClick systems so there would be no service disruptions). Google's Third Set of Interrogatory Responses, June 30, 2025 ("The rewrite of DART into what became known as "DoubleClick for Publishers" ("DFP") and the migration of DART's customers to DFP took place over nearly seven years ... [a]pproximately 50 software engineers worked on rewriting DART, while 50 software engineers maintained the legacy DART systems to ensure there would be no service disruptions for DoubleClick's customers."). By September 2011, roughly 3 years into the process of migrating DoubleClick, only 45% of publisher volume had migrated over. And Google observed that, "to upgrade the other 55% we're going to have to scale our upgrade process significantly," GOOG-AT-MDL-012318515 at -515, which eventually took over 3 more years.

2008.[454] Thus, enabling a divested AdX or DFP to be able to scale to Google's current size would be akin to asking the SWEs to replicate many years' worth of Google's research, experience, and lessons learned on scaling technology.

190. I would expect that substantial changes to AdX's or DFP's software would be required to ensure that it can operate at the scale it does today, but on the new infrastructure. Google's proprietary infrastructure on which GAM's software currently runs enables GAM's software to scale in a way that avoids GAM's engineering team having to worry about whether the product will cease to function or encounter significant performance issues. That benefit is lost once the divested product is moved outside Google infrastructure.

191. For example, as discussed previously, Colossus provides data center storage management and handles storage disk failures, so that Spanner can focus on being a globally distributed database with a robust data consistency model without worrying about storage issues, and F1 in turn does not concern itself with global reliability and can therefore focus on being a highly efficient relational database management system. This means that, without Google's proprietary infrastructure, AdX or DFP software would not benefit from those underlying guarantees, such as Google's transparent and easy-to-use data consistency model. Thus, it is likely that SWEs would need to write new, highly complex code for the divested product's software and software infrastructure to maintain data consistency and enable horizontal scaling on a global level, which was previously addressed outside of the divested product by Google's proprietary supporting infrastructure.[455]

5. Migrating Customers

192. Even assuming SWEs are able to successfully migrate the AdX or DFP software to new infrastructure, confirm that it has the same features and capabilities that GAM's ad serving and exchange functionality have today, and verify that the software can scale to meet customers' needs, SWEs would then have to undertake the task of migrating customers to the product operating in the buyer's environment.

193. The time needed to migrate data is highly dependent on a number of factors—how much additional software tooling is needed to perform the actual data move, whether any steps in

[454] Google's Third Set of Interrogatories, June 30, 2025 ("Today, Google Ad Manager is many times larger and more complex than DART was. For example, in 2023, Google Ad Manager serviced approximately 65 times more ad requests than DART did in 2007 and 2008, and numerous features have been added to DFP and subsequently Google Ad Manger's ad serving functionality since the initial migration from DART.").

[455] This similarly applies to the other supporting software and infrastructure for DFP, such as Google's infrastructure and software for running experiments. Without Mendel and RASTA, engineers for DFP would not have an easy-to-use experiment system that scales horizontally and globally while ensuring rapid turnaround and results analysis and reporting.

the migration efforts initially fail (which is quite common in migrations[456]), the duration and causes of any occurrences of customer data outages or service disruptions, and how much data there is to transfer.

194. To perform a migration with minimal disruption to customers, at a minimum, SWEs would need to operate and maintain both the previous and new operating environments. This is necessary to support both Google's customers who have not been migrated to the divestiture buyer, and those customers who have been migrated.[457]

195. SWEs would also need to build tools to duplicate traffic to both the previous and new operating environments for testing purposes and compare the results of the DFP software running in the new environment with the results of the DFP software running in the previous Google infrastructure environment. This would allow SWEs to validate that the DFP software running in the new environment produces the same results as in the previous Google environment, and validate that DFP in the new environment can operate as it did before. As to AdX, it is likely that at least a fraction of AdX's volume would need to be migrated first to the divestiture buyer's auction systems while those systems are still under development and scaling up so that the performance of the new systems could be tested (with the remaining volume still being held on Google's old AdX infrastructure). Duplicating output data to both old and new systems is a standard practice when preparing to migrate software architectures to new systems. This is easier when architecture migrations are being performed within a single company. Doing so between two separate companies in the proposed divestiture scenario raises several complications.

196. Divestiture introduces the additional problem that the old and new systems reside with two different entities: Google and the divestiture buyer. Private customer and user data thus must cross over between the two companies' infrastructures. This would likely complicate the already novel and presumably custom-built migration code that would be required to ensure a smooth migration. It is also another reason to believe that timelines from prior migrations are likely to underestimate the time and resources likely to be required here.

197. SWEs would also need to develop mechanisms to enable the actual migration of customer data to the new environment. This could be complicated by differences in database

[456] For example, in the course of migrating and integrating Mandiant into GCP, Google engineers repeatedly encountered issues that required more research and help, leading to either changing the initial plan or waiting for additional work to be done to fix the issue. *See* GOOG-AT-MDL-B-009835903, at -910 ("In the course of working through the challenges of standing up Cloud Ferry and related development enablement at Google there's a recurring pattern that follows a sequence of: … Hit a gap that is at least partially blocking. … Research and/or ask for help/suggestions on that topic. … Change plan or wait for fix to gap. … Move on.").

[457] GOOG-AT-MDL-004216796, at -798 ("While the new platform [XFP] is being rolled out in phases, the DFP [XFP] and DART systems continue to run in parallel and satisfy the business needs of legacy and new publishers.").

schemas[458] that may arise from differences in how the underlying database infrastructures operate in the new environment versus the previous environment. Such complications could significantly increase the time required for data migration. Even aside from these potential complications, the time required to transfer such an enormous amount of data—i.e., the literal transfer of bytes of data from one system to another—would take a considerable amount of time given the volume of data involved.

198. An important requirement and caveat to consider during the migration process (as well as the rest of the divestiture process) is how to minimize (and eliminate as much as possible) service disruptions for all stakeholders.[459] The customer data migration would necessarily involve placing additional load on the old versions of the products and existing infrastructure, thus increasing the likelihood of disruptions even if stakeholders were continuing to use the old systems. Some possible sources of service disruptions during customer data migration that would impact publishers and potentially jeopardize their ongoing use of the products include: data loss, data corruption, system and infrastructure downtime due to unexpected bugs or errors,[460] old or "stale" data being used, degradation in the quality of features due to data issues or system issues, and slow performance. Customer support for the new version of AdX or DFP may also be less than ideal due to lack of

---

[458] A database schema is the overall structure or organization of data within a database. For example, a table structure for data.

[459] Publishers have repeatedly testified in this case about the work and disruption involved in any ad server migrations. *E.g.*, Wolfe Trial Testimony (Gannett), 9/9/2024 at 71:3-25 ("[Q.] Could you tell the Court a little bit about that process. … [A.] Sure. In -- I guess in 2010 or so, we started the evaluation. In 2011, we moved from an instance of ad tech helios, which was our previous ad server, to Google's DoubleClick for Publishers product. … [Q.] Could you tell the Court a little bit about what that process actually entailed? … [A.] The entire process -- the entire process is a very heavy lift for any publisher, especially someone of our size in terms of the number of websites and properties. It did take us approximately a year. I believe we started in November 2011 and ended in November of 2012 with the migration. But, again, it's such a heavy lift, you know, it's akin to, you know, changing the tires on the race car mid race. It's not like we can, you know -- we have to continue to support all of the ad sales that we have in flight while making adjustments to not only the websites, but making further adjustments in integration to upstream technologies, customer relationship management tools, order management tools, as well as downstream financial and reporting tools. So it is an incredibly laborious lift."); Matthew Wheatland Deposition (Daily Mail), 6/24/2025 at 222:23-223:17 ([redacted]); Layser Trial Testimony (former News Corp, current Amazon), 9/10/2024 at 27:13-28:3 ("So there would be switching costs, so you would have to move from one ad–like, you would have to move ad campaigns from one ad server to the other. You would also need to, like, take kind of time from all of your different teams who are implementing the new ad server to make sure that things were set up correctly, to launching things. There's risk that's associated with moving from one ad server to another.").

[460] Glenn Berntson Deposition, 6/24/25, at 16:9-22:12 ("[T]here are lots of examples of where migrations have resulted in degradation of performance -- have the chance of introducing bugs, for example, that would result in ads not being effectively delivered and impacting a publisher's business ...").

experience with the new systems among both SWEs and customer support specialists. Contingencies such as these during the migration process must be accounted for and proactively addressed, which will take significant planning and could lead to additional delays throughout the divestiture process.

6. Optimizing System Performance To Attempt To Achieve Performance Parity to Today's Google Ad Manager

199. Even assuming SWEs are able to migrate all user data and validate the reliability of the new divested product, it is likely that further adjustments to optimize the system would be required as the buyer's SWEs gain experience working with the system. A significant amount of optimization efforts would be necessary up front to iron out transition issues and learn more about the real-world operation of the divested system—all of which is needed to run the divested product at the performance levels it previously did.[461]

200. The optimization process may identify code or design issues that must be addressed through code optimization, a total software or system redesign, or something in between. For example, when Google acquired DoubleClick and rewrote the then-existing DoubleClick publisher ad server on Google infrastructure as XFP, Google engineers identified key optimization objectives for XFP and dedicated a team of engineers to focus on those objectives.[462]

201. As another example of optimization work the buyer may need to do, Google has dedicated monitoring systems that identify slow-running, low-performance portions of software code

[461] Nitish Korula Deposition, 6/9/25, at 146:13-151:1 ("Like, in the example of EDA that I mentioned before … you have something running, but actually to feel like this is good -- feel like this is actually good for customers, this is going to take considerably longer, and you would run experiments … we need to find bugs, you fix the bugs, all of this stuff is -- it takes years to even gather the data to know that something is wrong maybe."); Tim Craycroft Deposition, 6/25/2025, at 82:11-84:1 ("[redacted]).

[462] GOOG-DOJ-13241698, at -721 ("xFP Optimization Q3 OKRs [Objectives and Key Results] … Identify optimization objectives for xFP … (e.g. scheduling, performance, revenue, etc.). … Finalize core decision logic w/ Ad Server team … scheduling/pacing algorithm … optimized ad delivery … include hooks for optimization even if it's not implemented in v0. …").

and hardware that act as bottlenecks.[463] Once the monitor identifies these cases, Google engineers profile the offending code sections with custom Google tools[464] such as Pprof (which is used to profile individual Google software and programs) and determine how to optimize them.[465] An example of code optimization would be identifying a portion of code that uses slower but less expensive computer memory (RAM), and modifying the code to instead use faster but more expensive memory caches (which are generally considered a limited resource). Likewise, the divestiture buyer would need to implement monitoring software that profiles the new DFP or AdX system and infrastructure, and dedicate engineers to optimizing the slow portions. Ideally, much of the groundwork for the monitoring system would have been developed during the earlier software migration stage when testing the scalability of the migrated software and new operating environment.

202. Google also employs monitoring systems such as Google-wide Profiling ("**GWP**")[466] that measure the overall performance of multiple cohesive services to identify bottlenecks at various levels of the software services, and not just at the code level. Google dedicates SREs and SWEs to the building and maintenance of monitoring infrastructure and systems that help other SWEs optimize other Google systems.[467] In complex systems, performance issues are not always obviously identifiable before the system is online and running, and monitoring systems are often necessary to provide sufficient information to help pinpoint performance bottlenecks while the system is already running. The buyer would thus require additional time, on top of building a new divested product, to also develop and maintain such monitoring infrastructure and systems.

203. Google SWEs also heavily depend on Mendel and RASTA to continuously evaluate features to identify areas for improvement for GAM, generally through A/B testing on live traffic at

[463] Borg also has built-in monitoring that is supplemented by other monitoring systems. *See* Abhishek Verma et al., "Large-Scale Cluster Management at Google with Borg," *Proceedings of the Tenth* European *Conference on Computer Systems (EuroSys)*, ACM (April 2015) at 4 ("Almost every task run under Borg contains a built-in HTTP server that publishes information about the health of the task and thousands of performance metrics (e.g., RPC latencies). Borg monitors the health-check URL and restarts tasks that do not respond promptly or return an HTTP error code. Other data is tracked by monitoring tools for dashboards and alerts on service level objective (SLO) violations.").

[464] GOOG-AT-MDL-B-009828860, at -864-65 ("Performance Analysis Tools …").

[465] GOOG-AT-MDL-B-009828860, at -864-65 ("Performance Analysis Tools … pprof … pprof is a profiling tool that allows you to collect various types of profiling data from running binaries (eg CPU, heap utilization, contention, etc). and save it in a file for further analysis and visualization.").

[466] *See* GOOG-AT-MDL-C-000018442 at -442 for Google-wide Profiling (GWP) ("From randomly-selected machines, it collects 10 seconds worth of machine-wide profiles, as well as 10-second CPU, memory, lock and thread profiles for each server. … Get a global picture of how CPU cycles are actually spent on Google production machines. … Get a global profile of a particular binary from different machines over a designated time period. … Identify hot functions across all Google3 applications, which may not appear hot in each individual application. …").

[467] *See generally* GOOG-AT-MDL-C-000018725 for Sawmill and ULS as examples of Google systems with dedicated SRE support teams.

scale.[468] In addition to this, Mendel and RASTA allow GAM to evaluate or gradually roll out new features to help manage risk that a new feature has inadvertent negative impacts on customers.[469]

204. Being able to rely on Mendel and RASTA significantly eases development processes for SWEs. Mendel abstracts away the difficulties of performing A/B testing on a global scale, while still supporting a high amount of granularity[470] despite the extensive rollout process of an experiment that happens behind the scenes.[471] RASTA abstracts away the remaining complexities of Mendel itself, due to analysis of Mendel experiments and their results requiring software engineering expertise, by providing an easy to use and understand user interface for analyzing and reporting on Mendel experiment data.[472] Together, Mendel and RASTA therefore allow Google SWEs to achieve a high level of control over experiments and A/B testing at a massive scale without needing to know the details of how Mendel and RASTA operate. To continue to optimize system performance, a divestiture buyer would need to recreate analogous systems in its own infrastructure set-up.

B. Prior Software Migrations, Including the Migration Of DoubleClick Onto Google Infrastructure, Have Taken Years Despite Being Less Complex Than Building a Divested DFP or AdX

205. In addition to evaluating the particular steps and challenges associated with creating a divested DFP or AdX, I have also evaluated that process in comparison to the complexity and time that Google has taken to complete other software migration projects—most notably for the migration of DoubleClick onto the Google technology stack.

[468] GOOG-AT-MDL-018528378, at -382 ("Why run experiments? … • Make decision based on experiment results ◦ revenue, click through rate, conversions …").
[469] GOOG-AT-MDL-018528378, at -382 ("Why run experiments? … • Manage risk ◦ roll out changes slowly ◦ roll back regressions fast …").
[470] GOOG-AT-MDL-018528378, at -393 ("Experiment configuration … • Traffic selection criteria ◦ Cookie ranges / request samplings ◦ Experiment conditions ◦ Diversion point ◦ Layers …"); at -394. ("• Conditions ◦ Additional filters based on properties of the request ◦ e.g. country, language, web property …").
[471] GOOG-AT-MDL-018528378, at -392 ("Life of an experiment … Wait for experiment to go to canary data center (~2 hours) … Wait for experiment to go global (~2 hours) …").
[472] *See generally* GOOG-AT-MDL-004017319 ("RASTA 101 for DRX PMs").

1. The DoubleClick Migration Took Nearly Seven Years for Products Much Less Complex Than They Are Today

a) Google Decided To Rebuild DFP And AdX To Operate Within Google's Bespoke Infrastructure

206. At the time of Google's acquisition of DoubleClick in 2008,[473] DoubleClick's primary ad serving product was referred to as DART for Publishers.[474,475] DoubleClick also operated an ad exchange known as DoubleClick Ad Exchange,[476] and an advertiser ad server referred to as DART for Advertisers.[477] DART for Publishers is not the same as Google's DoubleClick For Publishers ("DFP"), which refers to Google's rewrite of DART for Publishers.[478] And GAM refers to the Google product that contains ad serving functionality today, which evolved from DFP.

207. As part of the acquisition, Google decided that it should rewrite DART on Google infrastructure, for example by combining DART for Publishers and Google's then-existing ad server (coincidentally also named Google Ad Manager).[479] Similarly, Google also decided to rewrite DART for Advertisers to depend on Google storage backend infrastructure.[480] One motivating reason for the migration was that DART's performance

[473] GOOG-TEX-00806682, at -683 ("The deal to merge DoubleClick and Google was signed all the way back in April 2007 … the deal finally closed in March 2008.").

[474] GOOG-TEX-00552355, at -356 ("With DoubleClick, Google acquired the market leading publisher ad server, DART for Publishers (DFP)."). For clarity, in this report I refer to DART For Publishers as "DART," contrary to the nomenclature in around 2008-2009 from when this document was sourced.

[475] GOOG-AT-MDL-004216796, at -797 ("DoubleClick, when acquired by Google, had an existing publisher display ads product called DART for Publishers, externally known as DFP.").

[476] "DoubleClick Advertising Exchange," *doubleclick*, https://web.archive.org/web/20071031072045/http://www.doubleclick.com/products/advertisingexchange/, (Accessed June 7, 2025) ("The DoubleClick Advertising Exchange service makes buying and selling digital advertising faster, easier and more profitable. … © 2007 DoubleClick Inc.").

[477] GOOG-AT-MDL-004216796, at -798-99 ("DoubleClick for Advertisers (DFA) … DoubleClick's legacy display ad trafficking platform that depended on DART technology was known as DFA.").

[478] After Google acquired DoubleClick, Google engineers began work on a project codenamed "xFP" (or "XFP") that had the goal of combining the acquired DoubleClick DART for Publishers product with Google's GAM/GFP ad server product. *See* GOOG-DOJ-01430825, at -827 ("Decision: Build out XFP … Combine DFP and GAM on the Google stack."); *see also* GOOG-TEX-00552355, at -356 ("Both teams, DFP and GAM engaged quickly in the dialogue about how to develop the next generation puublisher [sic] ad server, combining the best of both worlds. The merged product, code named xFP, is now well under development."); GOOG-AT-MDL-004216796, at -798 ("Internally, GAM was known as GFP. When the companies merged, XFP, a new project that combined the strengths of DoubleClick (DART) and Google (GAM) technologies emerged.").

[479] GOOG-DOJ-01430825, at -827 and -830 ("Decision: Build out XFP … Combine DFP and GAM on the Google stack … Decision: Move DFP, DFA and DART Search reporting to the Google stack.").

[480] GOOG-DOJ-11999657, at -658 ("xFA API is the implementation of the DoubleClick for Advertisers [DART (DFA)] API in Google3 technology stack against a new storage system ([F1]). … Goals: … Migration of customer data from Oracle to [F1]/[S]panner to move off legacy Oracle DB in DCLK.").

lacked scalability, and it was written using technology that was not in line with Google's already existing technology stack.[481,482]

208. Specifically, in 2008, DoubleClick operated its own network of 17 data centers scattered across the world.[483] The backend systems that hosted and served DART's data and software were built on various off-the-shelf server computers and technologies including hardware and software from Microsoft (Windows),[484] Sun Microsystems, NetApp, and Oracle.[485] For its production database and data management needs, DART relied on Oracle's database product ("Oracle DB"),[486] a proprietary database management system ("DBMS") that required large, expensive servers to support large workloads.[487]

209. There were many technical and financial reasons why it made sense to do for Google to move DART to Google's own infrastructure. DART was incompatible with Google's infrastructure, so keeping DART separate would have required maintaining incompatible infrastructure. Google's infrastructure does not use Windows Server.[488] Maintaining incompatible infrastructure would impose ongoing and unnecessary financial and resource costs, and, given concerns about the scalability of DoubleClick's existing infrastructure,

[481] GOOG-DOJ-11999657, at -659 ("Legacy DFA [DART (DFA)] Architecture … DFA6 UI (ASP.NET/C#) … MediaVisor UI (JRun/Java) … Perseus UI (ASP.NET/C#) …").

[482] Google's Third Set of Interrogatory Responses, June 30, 2025 at 14 ("For a variety of reasons, including the costs and undesirability of maintaining separate data centers with incompatible software architectures as well as concerns regarding the scalability of DoubleClick's then-existing infrastructure, Google decided to rewrite DART to run on top of Google's software and hardware infrastructure and integrate it fully into Google's growing network of data centers.").

[483] GOOG-DOJ-01823992, at -009 ("DoubleClick - A Quick Profile … 17 data centers around the world.").

[484] Including versions of the Windows Server operating system. *See* Google's Third Set of Interrogatory Responses, June 30, 2025 at 13 ("At the time of its acquisition by Google, DoubleClick offered a publisher ad server known as "DART for Publishers" (for purposes of this response, "DART") and an ad exchange known as "DoubleClick Ad Exchange" ("AdX"), each of which was hosted on server infrastructure that was owned and operated by DoubleClick in a network of data centers around the world. Both products ran on top of a version of the Windows Server operating system and used Oracle as a database management system.").

[485] GOOG-AT-MDL-B-009836315, at -329 ("FSADS System … Fast WIN2000 file servers that hold multiple days worth of adsvr [ad server] logfiles … FSSIM System … Fast file servers [NetApps F760] that hold 3-5 days worth of sim files … CIFS & NFS File System … EPS System … Reads all sim file [sic] in real time sorting the data and aggregating it via Ab Initio. The 4 SUN e6500 [24 way] servers use a form of clustering for loading sharing. … Oracle DB … DCLK production database.").

[486] GOOG-AT-MDL-B-009836315, at -331 ("Current Ad Manage System … Oracle DB … DCLK production database …"); *see also* GOOG-DOJ-04292352, at -354 ("The XFA serving team has implemented an alternative serving system to DART … Similarly the XFA team has started replacing the DFA API server that is currently storing all its data in Oracle with one that is written on Google3 …"); GOOG-DOJ-11999657, at -658 ("xFA API is the implementation of the DoubleClick for Advertisers [DART (DFA)] API in Google3 technology stack against a new storage system ([F1]). … Goals: … Migration of customer data from Oracle to [F1]/[S]panner to move off legacy Oracle DB in DCLK."); GOOG-DOJ-11999657, at -660 ("Legacy [DART (DFA)] Architecture contd. … PROD (Oracle).").

[487] Oracle, "Database," *Oracle*, https://www.oracle.com/database/, (accessed July 6, 2025).

[488] Google, at large, uses custom versions of the Linux operating system. Google's Third Set of Interrogatory Responses, June 30, 2025 ("Google, by contrast, operated (and still operates) its own network of data centers using largely servers running customized versions of the Linux operating system.").

2. Other Ad Tech Software Projects Experienced Similar Challenges and Demonstrate the Complexity of Creating a Divested DFP or AdX

229. In my experience, there is little, possibly even no, precedent for the kind of project Plaintiffs contemplate—recreating all at once such a massive system as AdX or DFP, which is tailored to a single software infrastructure and operates on such a significant scale. However, like the migration of DoubleClick products onto Google's infrastructure, other software projects that Google embarked on for its ad tech products in the ordinary course of Google's business can still be probative of how difficult steps within the proposed divestiture might be, even if those migrations are not analogous to divestiture of AdX or DFP. As explained below, Google's experience with rewriting software systems in a more limited way—even ***within*** Google's internal infrastructure—demonstrates the significant technical challenges rewriting GAM code would encounter given the complexity of the code base underlying Google's ad tech products.

230. In late 2012, Google performed an internal migration of just publisher data from MySQL to F1—a part of Google's internal infrastructure—for both DFP and AdX.[534] Even though Project Agave involved migrating only publisher data within Google infrastructure at a time when Google's sell-side business was smaller, Project Agave still took around four to five years.[535] It demanded the efforts of more than 100 different Google engineer contributors.[536]

231. Project Grumpy was an internal Google effort to simplify the design of Google's ad serving system[537] that had begun by June 2013.[538] The further unification of the serving systems used by Google's different ad tech tools, including DFP and AdX, was expected to yield benefits such as improving the consistency of the system's behavior and of the user experience (e.g., eliminating discrepancies in reporting across different Google ad tech products), making it easier to depend on shared Google services (e.g., logging), simplifying cross-product interactions (e.g., connections between sellers and buyers, and easier cross-product testing), and reducing the need to redirect requests to external systems thereby

[534] GOOG-DOJ-13940968, at -968 ("Agave is an infrastructure effort started on DFP in late 2012"), and GOOG-DOJ-04430863, at -864 ("What is Agave? … Agave is a project to move all existing DRX data from the old database (AdsDB) using MySQL to a completely new F1 database (Agave).").

[535] GOOG-DOJ-13940968, at -971 ("DRX Agave Infrastructure … Timeframe: Q3 2012 - Q1 2017."). The Agave F1 migration for AdX took over 2 years. *See* GOOG-DOJ-13940968, at -968 ("AdX Agave F1 Migration … Timeframe: Q1 2015 - Q3 2017.") As of February 2017, Project Agave for GAM was planned to be completed and usable in Q3 2017; GOOG-DOJ-13209291, at -307 ("Progress on AdX Agave that will lead us to unified inventory, on track for early Q3 [2017].").

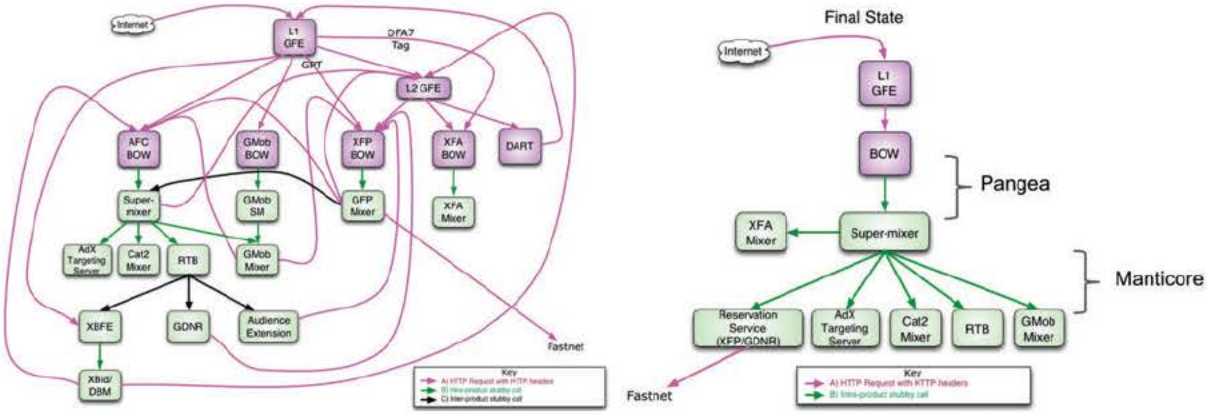
[536] GOOG-DOJ-13940968, at -972 ("Over several years the project [Project Agave] had 100+ eng[ineer] contributors and tracked 3000+ issues."). Project Agave's AdX migration alone required approximately 30 Google engineers. *See* GOOG-DOJ-13940968, at -969 ("I led a cross functional team of ~30 engineers to define the new AdX schema, as well as ensured the safe migration of systems to the DRX F1/Spanner database.").

[537] GOOG-DOJ-12000039, at -043 ("Why Grumpy? … Simpler, flexible, acyclic serving stack …").

[538] GOOG-DOJ-AT-01573572, at –572 ("Project Grumpy: Next Generation Display Ads Serving … http://go/project-grumpy … Last update not before: June 6, 2013.").

improving overall privacy.[539] Figure 14 depicts Project Grumpy's planned "before" and "after" states of Google's ad serving stack, highlighting the consolidation and simplification of the overall system that further intertwined the code for Google's various ad tech tools.

Figure 14: Project Grumpy Planned "Before" And "After"[540]



232. Project Grumpy was effectively completed as of late 2020 with the success of Project AURAS, which was itself an "offshoot" of Project Grumpy with the same goal of unifying systems.[541] From beginning to end, Project Grumpy took over seven years of effort across several teams,[542] despite being a "priority" for the relevant teams for years.[543]

233. Importantly, the software projects described above were challenging and took years even though they leveraged and were implemented ***within*** Google's already available, highly efficient and optimized, and transparent bespoke infrastructure, which did not itself need to be rewritten.[544] By contrast, the proposed divestitures of AdX and DFP would require not

[539] GOOG-DOJ-12000039 at -043 ("Why Grumpy? … Simplify interconnections between sellers and buyers … Testability of cross-product interactions Consistent behavior and user experience across multiple display products with a shared serving stack … Zero discrepancy reporting … Modularity, easier to share core services (pub filters, reservation, TYM, logging, mobile features, and more) … Privacy … significantly less external redirects.").

[540] GOOG-DOJ-12000039 at -042 and -044.

[541] GOOG-AT-MDL-B-005180709, at -709 ("Project AURAS launched its final milestone in 2020/Q2 (announcement) and the landing and cleanup work was completed in 2020/Q3. AURAS landing was announced in 2020/03, and this project is now complete. … Project AURAS is an effort to unify the auction and reservation serving stacks. It is the last, and perhaps largest, offshoot of project Grumpy which was focused on unifying various pieces of the display advertising serving stack within Google. …").

[542] GOOG-AT-MDL-011226177, at -178 ("And more than seven years after the initial Project Grumpy kick off to align and integrate all of the Display stacks, we can finally say that vision has been realized."

[543] GOOG-DOJ-13615596 at -597; GOOG-AT-MDL-008522254 at –254.

[544] Glenn Bernston Deposition, 6/25/25 at 98:15-100:11 (". . . a bunch of work I've done supporting AdX, including internal migrations for things that seem really basic, that actually take a really long time when you roll them out in a way to make sure that your partners, in this case, bidders, are not negatively impacted by the changes you're putting in place.").

just disentangling the systems that projects like Grumpy and AURAS unified, but more importantly writing new versions of AdX and DFP to work outside Google's infrastructure and, depending on what infrastructure is available to the buyers, rewriting the dependencies in the underlying infrastructure.

C. Divestiture Of Either AdX or DFP Would Highly Likely Require At Minimum Five Years Even in a Best-Case Scenario

234. Considering all of the steps and complexity required to create a divested AdX or divested DFP that could successfully operate at a level that would be equivalent to (i.e., have the same features and capabilities as) the AdX and DFP functionalities as they exist today, I estimate that any feature-equivalent divestiture of either AdX or DFP to a third-party buyer such that the third-party buyer can operate AdX or DFP outside Google infrastructure would be highly unlikely to be achieved in less than five years, even in a best-case scenario. Five years is an optimistic lower bound, and I cannot say with confidence that it is likely the project could be concluded in that time. To the contrary, it could take considerably longer than that, perhaps even twice as long, and the result might still not be an ad server or ad exchange with similar functional and scaling capabilities as Google's current product offering.[545] Moreover, because that best-case scenario would still involve each of the stages described above (and relies on the improbable absence of any complicating factors), I do not believe that increasing the number of SWEs beyond the 200-300 I have assumed would be available for the project would reduce the time required below that lower bound.

235. During the entire time period described above, the engineering team that works on sell-side functionality with expertise in Google's ad technology would necessarily be diverted away from both maintaining the legacy product and from improving on either the legacy or divested products. There is a trade-off between fixes of bugs on an existing product, product improvements, and the enormous task of divestiture.[546] A diversion of head count would directly decrease Google's ability to make investments into its products by delaying—or entirely ceasing—urgent bug fixes, code simplification, improved publisher troubleshooting, infrastructure migration, adapting to privacy changes, and long term improvements like extension of products to keep up with shifts in digital content user

[545] *See also* Tim Craycroft Deposition, 6/25/25, at 78:11-18 ([redacted]).

[546] George Levitte Deposition, 6/13/25, at 69:17-71:5 ([redacted]").

behavior.[547,548] Additionally, a divestiture and migration adds significant uncertainty for publishers who depend on their business' sophisticated and complex set-ups, including software and API dependencies that are tailored to GAM,[549] and who depend on the high performance of GAM's systems operating on Google infrastructure.[550]

236. I further note that, though I have assumed for purposes of this report that the divested DFP and AdX must have equivalent (or as close as possible) features and scaling capabilities as GAM's ad serving and exchange functionality do today, it may be impossible to achieve perfect feature equivalence within any reasonable amount of time (even within the time frames described in this report). That is because parts of GAM's existing functionality currently benefit from Google's other products[551] in ways that may not be replicable in a divested product.

237. As one final consideration, it is well known in the software engineering field that the larger a project is, the more difficult it is to accurately predict the required time and resources for completion. This is because as a project grows, it becomes exponentially more complex, making it harder to account for all known contingencies, on top of it being impossible to foresee the unknown ones. Because predicting timelines is hard and divesting AdX or DFP would be a massive undertaking, I therefore approach this section with the caveats that the proposed lower bound time estimate is measured in years with a wide range, and that my estimates are conservative lower bounds given that timelines for complex engineering

[547] GOOG-AT-MDL-B-009832007, at -007, 009-010 [redacted]).

[548] Glenn Berntson Deposition, 6/24/25, at 22:13-24:11 ("If what we're talking about is a process where GAM as a whole has to be rebuilt someplace else, outside of Google, for example, that would be a huge amount of work. … So now you have sort of this case where resources are being pulled out to work on this other effort that could have a direct impact on our ability to provide innovations that themselves, are critical for these publishers to run their businesses.").

[549] Glenn Berntson Deposition, 6/24/25, at 16:9-22:12 ("[T]he way ads work, one of the key places of integration for web publishers is they'll integrate our software development kit, and SDK, into their website. And that SDK had what we would call, basically, an API, an application programming interface. Where the way a publisher configures how they want an ad to serve, or attributes of the page, are all very specific, say, parameters that you would set within an ad request, using the SDK. When you make changes to something like an SDK where you then change those APIs, it has a direct impact on the publisher, because they have to go back and change what their integration looks like. … So the deep integration that exists today, with the reliance on a system that is highly reliable and highly performant, is how publishers have built their businesses. … And when we disrupt those processes, there's a direct impact on publishers, in terms of work that they have to do to comply, or an actual degradation in terms of how effective GAM would be as a monetization platform for them.").

[550] Glenn Berntson Deposition, 6/24/25, at 24:13-26:21 ("So, fundamentally, the dependence and integration and sophistication of the services that are provided by GAM are all pretty much at risk in terms of a degradation of performance and what publishers need to be able to run their business, all the way from support, to performance, to scalability, to core functionality; whether that's purely based on performance, innovation on their side that needs to be supported, or even regulatory.").

[551] Nitish Korula Deposition, 6/9/25, at 142:2-146:4 ("And so there's a bunch of things of varying levels of importance that you couldn't perfectly replicate outside Google systems," including examples that rely on functionality developed for Google Search and AI capabilities).

projects routinely fail to predict complexities that arise once the engineering project begins.[552] To the best of my knowledge, no software migration of the scale and complexity proposed by Plaintiffs has ever been attempted. Any divestiture of AdX or of DFP would therefore be unprecedented, have uncertain timelines, and entail an unknown probability of success.[553]

VI. PLAINTIFFS' PROPOSED OPEN-SOURCE FINAL AUCTION WOULD POSE SIGNIFICANT TECHNICAL CHALLENGES

238. I understand that, as the "second phase" of their requested DFP divestiture, Plaintiffs propose to require Google to (1) "separate out the functionality that performs the final auction within the publisher ad server (*i.e.* the "auction logic" that determines which ad to render on the page from the available direct-sold opportunities and bids from indirect sources)", (2) "provide [that] code under an open-source license to industry organizations or participants," and (3) "provide an API. . . to allow a neutral, third-party industry organization (or another entity chosen by the publisher) to use the open-source code to administer the final auction outside of Google's control."[554]

239. There are, however, multiple aspects of that proposal that are ambiguous or undefined, including but not limited to what logic would constitute the "final auction"[555,556] and how the

[552] Nitish Korula Deposition, 6/9/25, at 55:16-56:17 ("Usually, considerably more often, they took more time than estimated.").

[553] *See also* Tim Craycroft Deposition, 6/25/25, at 75:16-24 ("[redacted]").

[554] Dkt. 1483-1 at 10-11.

[555] *See, e.g.*, Nitish Korula Deposition, 6/9/25, at 303:1-305:4 ("And if that is literally what you are saying the final auction logic is, then sure, yes, you could outsource that thing in the very limited case that you have. Not outsource – but separate that thing in the very limited case that you have a single ad slot and you don't have any of this other complexity. But I -- I presume, because of the trivial nature of the ad sell, that that's not actually what you meant. And that's kind of what I was apologizing for. The way that question makes sense to me, it seems like a very trivial question, such that I'm not sure I'm interpreting what you have meant."), 308:8-310:10 ("Here it's saying the final auction logic involves basically one of the core functionalities of an ad server, right. So, again, if we're going to -- sure, you separate all of the core functionalities of an ad server we can talk about that. But saying -- pulling out just the final auction logic is a – it's not a very meaningful thing.").

[556] Tim Craycroft Deposition, 6/25/25, at 280:22-281:12 ("Q. In what circumstances is enhanced dynamic allocation part of the final auction logic? . . . A. It really just depends on how you want to draw a box around the auction logic, so it's sort of meaningless semantics. In the end, enhanced dynamic allocation computes a modified bid for a reservation campaign based on how well that campaign is tracking to its contractual commitment to deliver a budget over a certain amount of time, and so it may construct a synthetic bid that's higher to ensure the publisher lives up to their contractual obligation.").